UNIVERSITY OF CALIFORNIA SAN DIEGO

Maxing, Ranking and Preference Learning

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Electrical Engineering (Machine Learning and Data Science)

by

Venkatadheeraj Pichapati

Committee in charge:

Professor Alon Orlitsky, Chair
Professor Kamalika Chaudhuri
Professor Tara Javidi
Professor Daniel Kane
Professor Young-Han Kim

2019

The dissertation of Venkatadheeraj Pichapati is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____

Chair

University of California San Diego

2019

DEDICATION

To my parents and sister.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I thank my parents Amrutha and Rami Reddy and my sister Silpa for their constant love, support and guidance. I also appreciate my nephews Nihal and Dhanvin's constant reminders to graduate.

I would like to thank my friends Somok Mondal, Chaitanya Ryali, Sai Kumar Arava, Tejaswy Pailla, Anusha Lalitha, Sankeerth Rao, Shouvik Ganguly, Alankrita Bhatt, and many more for their constant support. I will always cherish the memories I made with them. I also like to thank my tennis and trivia partners for all the fun times with them.

In UC San Diego, I was fortunate to have interacted with great teachers and mentors. I am thankful to my committee members Prof. Kamalika Chaudhuri, Prof. Young-Han Kim, Prof. Tara Javidi, and Prof. Daniel Kane.

I want to specially thank my previous colleague Ananda Suresh Theertha who helped me in joining the research group, finding my dissertation topic and getting internship opportunities. I want to thank my other current and previous colleagues: Jayadev Acharya, Yi Hao, Ashkan Jafarpour, Ayush Jain, Sudeep Kamath and Vaishakh Ravindrakumar for all the technical and non-technical discussions. Their collaborations made my stay at lab fun and exciting. I would also like to acknowledge my colleague and friend, Moein Falahatgar for his company in research, travelling and tennis.

My sincerest acknowledgements will go to my advisor Prof. Alon Orlitsky for the deep and fruitful discussions. I have always admired his approchability and pursuit of simplicity and elagance and I will strive to incorporate them in my future research. This dissertation wouldn't have been possible without his broad knowledge yet clear insights.

Chapter 2, in full, is a reprint of the material as it appears in *International Conference on Machine Learning*. Falahatgar, Moein, Alon Orlitsky, Venkatadheeraj Pichapati, and Ananda Theertha Suresh, 2017. The dissertation author was the primary investigator and author of this paper.

Chapter 3, in full, is a reprint of the material as it appears in *Advances in Neural Information Processing Systems*. Falahatgar, Moein, Yi Hao, Alon Orlitsky, Venkatadheeraj Pichapati, and Vaishakh Ravindrakumar, 2017. The dissertation author was the primary investigator and author of this paper.

Chapter 4, in full, is a reprint of the material as it appears in *International Conference on Machine Learning*. Falhatagar, Moein, Ayush, Jain, Alon Orlitsky, Venkatadheeraj Pichapati, and Vaishakh Ravindrakumar, 2018. The dissertation author was the primary investigator and author of this paper.

Chapter 5, in full, has been submitted for the publication of the material as it may appear in *Advances in Neural Information Processing Systems*. Falahatgar, Moein, Alon Orlitsky, and Venkatadheeraj Pichapati, 2019. The dissertation author was the primary investigator and author of this paper.

VITA

2011	B.Tech. in Electrical Engineering, Indian Institute of Technology, Kanpur
2016	M.S. in Electrical Engineering (Communication Theory and Systems), University of California San Diego
2019	Ph. D. in Electrical Engineering (Machine Learning and Data Science), University of California San Diego

PUBLICATIONS

Pichapati, Venkatadheeraj, and Parul Gupta. "Practical considerations in cluster design for co-ordinated multipoint (CoMP) systems." *IEEE International Conference on Communications (ICC)*, pp. 5860-5865. IEEE, 2013.

Devi, UmaMaheswari, Hariharasudhan Viswanathan, Ravi Kokku, Venkatadheeraj Pichapati, and Shivkumar Kalyanaraman. "On the estimation of available bandwidth in broadband cellular networks." *2014 Eleventh Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pp. 19-27. IEEE, 2014.

Pichapati, Venkatadheeraj, Hemant Kowshik, Anand Prabhu Subramanian, Ravi Kokku, and Malolan Chetlur. "Location assisted handoffs in dense cellular networks." *2014 Eleventh Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pp. 1-9. IEEE, 2014.

Falahatgar, Moein, Ashkan Jafarpour, Alon Orlitsky, Venkatadheeraj Pichapati, and Ananda Theertha Suresh. "Universal compression of power-law distributions." *2015 IEEE International Symposium on Information Theory (ISIT)*, pp. 2001-2005. IEEE, 2015.

Falahatgar, Moein, Ashkan Jafarpour, Alon Orlitsky, Venkatadheeraj Pichapati, and Ananda Theertha Suresh. "Faster algorithms for testing under conditional sampling." *Conference on Learning Theory*, pp. 607-636. 2015.

Kamath, Sudeep, Alon Orlitsky, Dheeraj Pichapati, and Ananda Theertha Suresh. "On learning distributions from their samples." *Conference on Learning Theory*, pp. 1066-1100. 2015.

Falahatgar, Moein, Ashkan Jafarpour, Alon Orlitsky, Venkatadheeraj Pichapati, and Ananda Theertha Suresh. "Estimating the number of defectives with group testing." *IEEE International Symposium on Information Theory (ISIT)*, pp. 1376-1380. IEEE, 2016.

Falahatgar, Moein, Alon Orlitsky, Venkatadheeraj Pichapati, and Ananda Theertha Suresh. "Learning markov distributions: Does estimation trump compression?." *IEEE International Symposium on Information Theory (ISIT)*, pp. 2689-2693. IEEE, 2016.

Falahatgar, Moein, Mesrob I. Ohannessian, Alon Orlitsky, and Venkatadheeraj Pichapati. "The power of absolute discounting: all-dimensional distribution estimation." *Advances in Neural Information Processing Systems*, pp. 6660-6669. 2017.

Falahatgar, Moein, Alon Orlitsky, Venkatadheeraj Pichapati, and Ananda Theertha Suresh. "Maximum selection and ranking under noisy comparisons." *Proceedings of the 34th International Conference on Machine Learning*-Volume 70, pp. 1088-1096. JMLR. org, 2017.

Falahatgar, Moein, Yi Hao, Alon Orlitsky, Venkatadheeraj Pichapati, and Vaishakh Ravindrakumar. "Maxing and ranking with few assumptions." *Neural Information Processing Systems*, pp. 7060-7070. 2017.

Falahatgar, Moein, Ayush Jain, Alon Orlitsky, Venkatadheeraj Pichapati, and Vaishakh Ravindrakumar. "The limits of maxing, ranking, and preference learning." *International Conference on Machine Learning*, pp. 1426-1435. 2018.

Yi, H. A. O., Alon Orlitsky, and Venkatadheeraj Pichapati. "On learning markov chains." *Neural Information Processing Systems*, pp. 648-657. 2018.

Falahatgar, Moein, Mesrob I. Ohannessian, Alon Orlitsky, and Venkatadheeraj Pichapati. "Towards Competitive N-gram Smoothing." *Submitted to Neurips*, 2019.

Falahatgar, Moein, Alon Orlitsky, and Venkatadheeraj Pichapati. "One Interview is Enough!: Sequential and Competitive Maximization." *Submitted to Neurips*, 2019.

Falahatgar, Moein, Ashkan Jafarpour, Alon Orlitsky, Venkatadheeraj Pichapati, and Ananda Theertha Suresh. "Universal compression of power-law distributions." *In preparation*.

ABSTRACT OF THE DISSERTATION

Maxing, Ranking and Preference Learning

by

Venkatadheeraj Pichapati

Doctor of Philosophy in Electrical Engineering (Machine Learning and Data Science)

University of California San Diego, 2019

Professor Alon Orlitsky, Chair

PAC maximum selection (maxing) and ranking of n elements via random pairwise comparisons have diverse applications and have been studied under many models and assumptions. We consider (ϵ, δ)-PAC maxing and ranking using pairwise comparisons for general probabilistic models. We present a comprehensive understanding of three important problems in PAC preference learning: maxing, ranking, and estimating *all* pairwise preference probabilities, in the adaptive setting.

SST + STI: We consider (ϵ, δ)-PAC maximum-selection and ranking using pairwise comparisons for general probabilistic models whose comparison probabilities satisfy *strong stochastic transitivity (SST)* and *stochastic triangle inequality (STI)*. Modifying the popular

knockout tournament, we propose a simple maximum-selection algorithm that uses $O\left(\frac{n}{\varepsilon^2}\log\frac{1}{\delta}\right)$ comparisons, optimal up to a constant factor. We then derive a general framework that uses noisy binary search to speed up many ranking algorithms, and combine it with merge sort to obtain a ranking algorithm that uses $O\left(\frac{n}{\varepsilon^2}\log n(\log\log n)^3\right)$ comparisons for $\delta=\frac{1}{n}$, optimal up to a $(\log\log n)^3$ factor.

SST +/- STI and Borda: With just one simple natural assumption: *strong stochastic transitivity (SST)*, we show that maxing can be performed with linearly many comparisons yet ranking requires quadratically many. With no assumptions at all, we show that for the Borda-score metric, maximum selection can be performed with linearly many comparisons and ranking can be performed with $O(n\log n)$ comparisons.

General Transitive Models With just *Weak Stochastic Transitivity (WST)*, we show that maxing requires $\Omega(n^2)$ comparisons and with slightly more restrictive *Medium Stochastic Transitivity (MST)*, we present a linear complexity maxing algorithm. With *Strong Stochastic Transitivity (SST)* and *Stochastic Triangle Inequality (STI)*, we derive a ranking algorithm with optimal $O(n\log n)$ complexity and an optimal algorithm that estimates all pairwise preference probabilities.

Sequential and Competitive We extend the well-known *secretary problem* to a probabilistic setting, and apply the intuition gained to derive the first query-optimal sequential algorithm for probabilistic-maxing. Furthermore, departing from previous assumptions, the algorithm and performance guarantees apply even for infinitely many items, hence in particular do not require a-priori knowledge of the number of items. The algorithm has linear complexity, and is optimal also in the streaming setting and for both traditional- and dueling-bandits. In a non-streaming setting, a modification of the algorithm is *competitive* in that it requires essentially the lowest number of queries not just in the worst case, but for every underlying distribution.

Chapter 1

Introduction

Maximum selection (maxing) and *sorting (ranking)* are two of the most fundamental and classical problems in Computer Science. These problems have been widely studied when the pairwise comparisons are non-noisy. As is well known , maxing requires n-1 comparisons, while ranking takes $\Theta(n \log n)$ comparisons.

In several real world applications, the pairwise comparisons produce only random outcomes. In sports, pairwise games with random outcomes are used to determine the best, or the order of teams or players. The popular crowd sourcing website GIFGIF [gif] shows how pairwise comparisons can help associate emotions with many animated GIF images. Other practical applications with random pairwise comparisons are in areas such as social choice [CN91, SCPX13], web search and information retrieval [RJ07, RKJ08] and recommender systems [BMR10].

The model of random pairwise comaprisons is also referred to as *dueling bandits*, a variation of *traditional multi-armed bandits* setting. Observe that under *traditional multi-armed bandits* setting, one can pull an arm and observe the reward where as under *dueling bandits* setting, one can pull two different arms and can only observe which arm gives the higher reward.

In this dissertation, we study the problems of maxing and ranking under noisy pairwise comaprisons. We establish probabilistic models under which $O(n)$ complexity maxing

and $O(n \log n)$ complexity ranking is possible. For both problems, we derive simple optimal algorithms that can be easily deployed in practice.

1.0.1 Notation

When two elements i and j are compared, i is chosen with some unknown probability $p_{i,j}$ and j is chosen with probability $p_{j,i} = 1 - p_{i,j}$ (no ties). Repeated comparisons are independent of each other. Let $\tilde{p}_{i,j} = p_{i,j} - \frac{1}{2}$ reflect the additional probability by which i is preferable to j.

We specify the desired output via the (ε, δ)-PAC paradigm that requres the output to likely closely approximate the intended outcome. Specifically, given ε, $\delta > 0$, with probability $\geq 1 - \delta$, maxing algorithm must output an ε-maximum element i such that for all j, $\tilde{p}_{i,j} \geq -\varepsilon$. Similarly, with probability $\geq 1 - \delta$, the ranking algorithm must output an ε-ranking $r'(1), ..., r'(n)$ such that whenever $r'(i) > r'(j)$, $\tilde{p}_{i,j} \geq -\varepsilon$.

1.0.2 Previous Results

Several probabilistic models were considered, including the popular Bradley-Terry-Luce [BT52] and its Plackett-Luce generalizations [Pla75, Luc05]. Yet even for such specific models, the number of pairwise comparisons needed, or *sample complexity*, of maxing and ranking was known only to a within a $\log n$ factor.

1.1 Strong Stochastic Transitivity and Stochastic Triangle Inequality

We consider PAC maxing and ranking under models more general than Bradley-Terry-Luce. More specifically, we assume an unknown ordering $\succ$ among the elements in $\{1, ..., n\}$ such that if $i \succ j$, then $\tilde{p}_{i,j} \geq 0$. Further, we assume two natural properties satisfied by the

Plackett-Luce model, hold whenever $i \succ j \succ k$: *Strong Stochastic Transitivity (SST)*, $\tilde{p}(i,k) \geq \max(\tilde{p}(i,j), \tilde{p}(j,k))$, and *Stochastic Triangle Inequality (STI)*, $\tilde{p}(i,k) \leq \tilde{p}(i,j) + \tilde{p}(j,k)$.

Under this more general model, we derive optimal $O(n)$ complexity maxing and $O(n \log n)$ complexity ranking algorithms. Notice that we consider more general models and yet derive algorithms with better complexity.

1.2 No Stochastic Triangle Inequality

We investigate if *not-so-natural Stochastic Triangle Inequality* is necessary for $O(n)$ complexity maxing and $O(n \log n)$ complexity ranking algorithms. We show that PAC maxing is still possible with $O(n)$ complexity where as ranking will require $\Omega(n^2)$ complexity.

1.3 General Transitive Models

We consider transitive models more general than *Strong Stochastic Transitive* models and study if $O(n)$ complexity maxing is possible. More specifically, we consider *Weak Stochastic Transitivity (WST)*, which is the most basic transitive condition that states whenever $i \succ j$, $\tilde{p}_{i,j} \geq 0$; *Medium Stochastic Transitivity (MST)*, sitting in between WST and SST, assumes that whenever $i \succ j \succ k$, $\tilde{p}_{i,k} \geq \min(\tilde{p}_{i,j}, \tilde{p}_{j,k})$.

We show that maxing under WST requires $\Omega(n^2)$ comparisons and $O(n)$ complexity maxing is possible under MST establishing MST as the most general known model for which $O(n)$ PAC maxing is possible.

1.4 Sequential and Competitive Maxing

We investigate if $O(n)$ maxing is possible with yet more restrictions. We consider the problem where the alternatives are presented in a streaming fashion. Further we also consider

that the number of alternatives is not even known in advance as when alternatives keep streaming. We study this problem under both *traditional multi-armed bandits* and *dueling bandits* settings and establish models under which $O(n)$ PAC maxing is possbile. We also provide a variation of these maxing algorithms that are *competitive* in that we optimize its complexity not just in the worst case, but for every underlying ditribution.

1.5 Thesis organization

The rest of this thesis is organized as follows:

- In Chapter 2 we study PAC maxing and ranking under models that satisfy both SST and STI constraints.
- In Chapter 3 we study PAC maxing and ranking under more general models that need not satisfy STI constraints.
- In Chapter 4 we study PAC maxing, ranking and preference learning under the most general transitive models.
- In Chapter 5 we derive optimal sequential and competitive maxing algorithms.

Chapter 2

Maximum Selection and Ranking under Noisy Comparisons

2.1 Introduction

2.1.1 Background

Maximum selection and sorting using pairwise comparisons are computer-science staples taught in most introductory classes and used in many applications. In fact, sorting, also known as *ranking*, was once claimed to utilize 25% of all computer cycles, *e.g.*, [Muk11].

In many applications, the pairwise comparisons produce only random outcomes. In sports, tournaments rank teams based on pairwise matches whose outcomes are probabilistic in nature. For example, Microsoft's *TrueSkill* [HMG06] software matches and ranks thousands of Xbox gamers based on individual game results. And in online advertising, out of a myriad of possible ads, each web page may display only a few, and a user will typically select at most one. Based on these random comparisons, ad companies such as Google, Microsoft, or Yahoo, rank the ads' appeal [RJ07, RKJ08].

These and related applications have brought about a resurgence of interest in maximum

selection and ranking using noisy comparisons. Several probabilistic models were considered, including the popular Bradley-Terry-Luce [BT52] and its Plackett-Luce (PL) generalization [Pla75, Luc05]. Yet even for such specific models, the number of pairwise comparisons needed, or *sample complexity*, of maximum selection and ranking was known only to within a $\log n$ factor. We consider a significantly broader class of models and yet propose algorithms that are optimal up to a constant factor for maximum selection and up to $(\log\log n)^3$ for ranking.

2.1.2 Notation

Noiseless comparison assumes an unknown underlying ranking $r(1),\ldots,r(n)$ of the elements in $\{1,\ldots,n\}$ such that if two elements are compared, the higher-ranked one is selected. Similarly for noisy comparisons, we assume an unknown ranking of the elements, but now if two elements i and j are compared, i is chosen with some unknown probability $p(i,j)$ and j is chosen with probability $p(j,i) = 1 - p(i,j)$, where if i is higher-ranked, then $p(i,j) \geq \frac{1}{2}$. Repeated comparisons are independent of each other.

Let $\tilde{p}(i,j) = p(i,j) - \frac{1}{2}$ reflect the *additional probability* by which i is preferable to j. Note that $\tilde{p}(j,i) = -\tilde{p}(i,j)$ and $\tilde{p}(i,j) \geq 0$ if $r(i) > r(j)$. $|\tilde{p}(i,j)|$ can also be seen as a measure of dissimilarity between i and j. Following [YJ11], we assume that two natural properties, satisfied for example by the PL model, hold whenever $r(i) > r(j) > r(k)$: *Strong Stochastic Transitivity (SST)*, $\tilde{p}(i,k) \geq \max(\tilde{p}(i,j),\tilde{p}(j,k))$, and *Stochastic Triangle Inequality (STI)*, $\tilde{p}(i,k) \leq \tilde{p}(i,j) + \tilde{p}(j,k)$.

Two types of algorithms have been proposed for maximum selection and ranking under noisy comparisons: non-adaptive or offline [RA14, NOS12, NOS16, JKSO16] where the comparison pairs are chosen in advance, and *adaptive* or *online* where the comparison pairs are selected sequentially based on previous comparison results. We focus on the latter.

We specify the desired output via the (ε,δ)-PAC paradigm [YJ11, SBFPH15] that requires the output to likely closely approximate the intended outcome. Specifically, given $\varepsilon,\delta > 0$, with

probability $\geq 1-\delta$, maximum selection must output an ε-*maximum* element i such that for all j, $p(i,j) \geq \frac{1}{2} - \varepsilon$. Similarly, with probability $\geq 1-\delta$, the ranking algorithm must output an ε-*ranking* $r'(1),\ldots,r'(n)$ such that whenever $r'(i) > r'(j)$, $p(i,j) \geq \frac{1}{2} - \varepsilon$.

2.1.3 Outline

In Section 2.2 we review past work and summarize our contributions. In Section 3.3 we describe and analyze our maximum-selection algorithm. In Section 2.4 we propose and evaluate the ranking algorithm. In Section 5.5 we experimentally compare our algorithms with existing ones. In Section 5.6 we mention some future directions.

2.2 Old and new results

2.2.1 Related work

Several researchers studied algorithms that with probability $1-\delta$ find the exact maximum and ranking. [FRPU94] considered a simple model where the elements are ranked, and $\tilde{p}(i,j) = \varepsilon$ whenever $r(i) > r(j)$. [BFHS14] considered comparison probabilities $p(i,j)$ satisfying the Mallows model [Mal57]. And [UCFN13, BFSH14, HSRW16] considered general comparison probabilities, without an underlying ranking assumption, and derived rankings based on Copeland- and Borda-counts, and random-walk procedures. As expected, when the comparison probabilities approach half, the above algorithms require arbitrarily many comparisons.

To achieve finite complexity even with near-half comparison probabilities, researchers adopted the PAC paradigm. For the PAC model with SST and STI constraints, [YJ11] derived a maximum-selection algorithm with sample complexity $O\left(\frac{n}{\varepsilon^2}\log\frac{n}{\varepsilon\delta}\right)$ and used it to bound the regret of the problem's dueling-bandits variant. Related results appeared in [SKS16]. For the PL model, [SBFPH15] derived a PAC ranking algorithm with sample complexity $O(\frac{n}{\varepsilon^2}\log n\log\frac{n}{\varepsilon\delta})$.

Deterministic adversarial versions of the problem were considered by [AFHN15], and by [AJOS14b, AFJ+16] who were motivated by density estimation [AJOS14a].

2.2.2 New results

We consider (ε, δ)-PAC adaptive maximum selection and ranking using pairwise comparisons under SST and STI constraints. Note that when $\varepsilon \geq \frac{1}{2}$ or $\delta \geq 1 - 1/n$ for maximum selection and $\delta \geq 1 - 1/n^2$ for ranking, any output is correct. We show for $\varepsilon < 1/4, \delta < \frac{1}{2}$ and any n:

- Maximum-selection algorithm with sample complexity $O\left(\frac{n}{\varepsilon^2}\left(1 + \log\frac{1}{\delta}\right)\right)$, optimal up to a constant factor.
- Ranking algorithm with $O\left(\frac{n}{\varepsilon^2}(\log n)^3 \log\frac{n}{\delta}\right)$ sample complexity.
- General framework that converts any ranking algorithm with sample complexity $O\left(\frac{n}{\varepsilon^2}(\log n)^x \log\frac{n}{\delta}\right)$ into a ranking algorithm that for $\delta \geq \frac{1}{n}$ has sample complexity $O\left(\frac{n}{\varepsilon^2}\log n(\log\log n)^x\right)$.
- Using the above framework, a ranking algorithm with sample complexity $O\left(\frac{n}{\varepsilon^2}\log n(\log\log n)^3\right)$ for $\delta = \frac{1}{n}$.
- An $\Omega\left(\frac{n}{\varepsilon^2}\log\frac{n}{\delta}\right)$ lower bound on the sample complexity of any PAC ranking algorithm, matching our algorithm's sample complexity up to a $(\log\log n)^3$ factor.

2.3 Maximum selection

2.3.1 Algorithm outline

We propose a simple maximum-selection algorithm based on Knockout tournaments. Knockout tournaments are used to find a maximum element under non-noisy comparisons.

Knockout tournament of n elements runs in $\lceil \log n \rceil$ rounds where in each round it randomly pairs the remaining elements and proceeds the winners to next round.

Our algorithm, given in KNOCKOUT uses $O\left(\frac{n}{\varepsilon^2}\left(1+\log\frac{1}{\delta}\right)\right)$ comparisons and $O(n)$ memory to find an ε-maximum. [YJ11] uses $O\left(\frac{n}{\varepsilon^2}\log\frac{n}{\varepsilon\delta}\right)$ comparisons and $O(n^2)$ memory to find an ε-maximum. Hence we get $\log n$-factor improvement in the number of comparisons and also we use linear memory compared to quadratic memory. From [ZC14] it can be inferred that the best PAC maximum selection algorithm requires $\Omega\left(\frac{n}{\varepsilon^2}\left(1+\log\frac{1}{\delta}\right)\right)$ comparisons, hence up to constant factor, KNOCKOUT is optimal.

[YJ11, SBFPH15] eliminate elements one by one until only ε-maximums are remaining. Since they potentially need $n-1$ eliminations, in order to appply union bound they had to ensure that each eliminated element is not an ε-maximum w.p. $1-\delta/n$, requiring $O(\log(n/\delta))$ comparisons for each eliminated element and hence a superlinear sample complexity $O(n\log(n/\delta))$.

In contrast, KNOCKOUT eliminates elements in $\log n$ rounds. Since in Knockout tournaments, number of elements decrease exponentially with each round, we afford to endure more error in the initial rounds and less error in the latter rounds by repeating comparison between each pair more times in latter rounds. Specifically, let b_i be the highest-ranked element (according to the unobserved underlying ranking) at the beginning of round i. KNOCKOUT makes sure that w.p. $\geq 1-\frac{\delta}{2^i}$, $\tilde{p}(b_i,b_{i+1}) \leq \varepsilon_i$ by repeating comparison between each pair in round i for $O\left(\frac{1}{\varepsilon_i^2}\log\frac{2^i}{\delta}\right)$ times. Choosing $\varepsilon_i = \frac{c\varepsilon}{2^{i/3}}$ with $c = 2^{1/3}-1$, we make sure that comparison complexity is $O\left(\frac{n}{\varepsilon^2}\left(1+\log\frac{1}{\delta}\right)\right)$ and by union bound and STI, w.p. $\geq 1-\delta$, $\tilde{p}(b_1,b_{\lceil\log n\rceil+1}) \leq \sum_{i=1}^{\lceil\log n\rceil+1}\frac{c\varepsilon}{2^{i/3}} \leq \varepsilon$.

For $\gamma \geq 1$, a relaxed notion of SST, called γ-*stochastic transitivity* [YJ11], requires that if $r(i) > r(j) > r(k)$, then $\max(\tilde{p}(i,j),\tilde{p}(j,k)) \leq \gamma\cdot\tilde{p}(i,k)$. Our results apply to this general notion of γ-stochastic transitivity and the analysis of KNOCKOUT is presented under this model. KNOCKOUT uses $O\left(\frac{n\gamma^4}{\varepsilon^2}\left(1+\log\frac{1}{\delta}\right)\right)$ comparisons.

Remark 1. *[YJ11] considered a different definition of* ε*-maximum as an element* i *that is at*

most ε *dissimilar to true maximum i.e., for* j *with* $r(j) = n$, $\tilde{p}(j,i) \leq \varepsilon$. *Note that this definition is less restrictive than ours, hence requires fewer comparisons. Under this definition, [YJ11] used* $O\left(\frac{n\gamma^6}{\varepsilon^2}\log\frac{n}{\varepsilon\delta}\right)$ *comparisons to find an* ε*-maximum whereas a simple modification of* KNOCKOUT *shows that* $O\left(\frac{n\gamma^2}{\varepsilon^2}\left(1+\log\frac{1}{\delta}\right)\right)$ *comparisons suffice. Hence we also get a significant improvement in the exponent of* γ.

To simplify the analysis, we assume that n is a power of 2, otherwise we can add $2^{\lceil \log n \rceil} - n$ dummy elements that lose to every original element with probability 1. Note that all ε-maximums will still be from the original set.

2.3.2 Algorithm

We start with a subroutine COMPARE that compares two elements. It compares two elements i, j and maintains empirical probability $\hat{p}_i$, a proxy for $p(i,j)$. It also maintains a confidence value $\hat{c}$ s.t., w.h.p., $\hat{p}_i \in (p(i,j) - \hat{c}, p(i,j) + \hat{c})$. COMPARE stops if it is confident about the winner or if it reaches its comparison budget m. It outputs the element with more wins breaking ties randomly.

Algorithm 1 COMPRARE

Input: element i, element j, bias ε, confidence δ.
Initialize: $\hat{p}_i = \frac{1}{2}$, $\hat{c} = \frac{1}{2}$, $m = \frac{1}{2\varepsilon^2}\log\frac{2}{\delta}$, $r = 0$, $w_i = 0$.

1. **while** ($|\hat{p}_i - \frac{1}{2}| \leq \hat{c} - \varepsilon$ and $r \leq m$)
   (a) Compare i and j. **if** i wins $w_i = w_i + 1$.
   (b) $r = r + 1$, $\hat{p}_i = \frac{w_i}{r}$, $\hat{c} = \sqrt{\frac{1}{2r}\log\frac{4r^2}{\delta}}$.

if $\hat{p}_i \leq \frac{1}{2}$ **Output:** j. **else Output:** i.

We show that COMPARE w.h.p., outputs the correct winner if the elements are well seperated.

Lemma 2. *If* $\tilde{p}(i,j) \geq \varepsilon$*, then*

$$Pr(\text{COMPARE}(i,j,\varepsilon,\delta) \neq i) \leq \delta.$$

Note that instead of using fixed number of comparisons, COMPARE stops the comparisons adaptively if it is confident about the winner. If $|\tilde{p}(i,j)| \gg \varepsilon$, COMPARE stops much before comparison budget $\frac{1}{2\varepsilon^2}\log\frac{2}{\delta}$ and hence works better in practice.

Now we present the subroutine KNOCKOUT-ROUND that we use in main algorithm KNOCKOUT.

Knockout-Round

KNOCKOUT-ROUND takes a set S and outputs a set of size $|S|/2$. It randomly pairs elements, compares each pair using COMPARE, and returns the set of winners. We will later show that maximum element in the output set will be comparable to maximum element in the input set.

Algorithm 2 KNOCKOUT-ROUND

Input: Set S, bias ε, confidence δ.
Initialize: Set $O = \emptyset$.

1. Pair elements in S randomly.
2. **for** every pair (i,j):

   Add COMPARE(i,j,ε,δ) to O.

Output: O

Note that comparisons between each pair can be handled by a different processor and hence this algorithm can be easily parallelized.

S can have several maximum elements. Comparison probabilities corresponding to all maximum elements will be essentially same because of STI. We define $\max(S)$ to be the maximum

element with the least index, namely,

$$\max(S) \stackrel{\text{def}}{=} S\Big(\min\{i : \tilde{p}(S(i), S(j)) \geq 0 \quad \forall j\}\Big).$$

Lemma 3. KNOCKOUT-ROUND(S, ε, δ) *uses* $\frac{|S|}{4\varepsilon^2} \log \frac{2}{\delta}$ *comparisons and with probability* $\geq 1 - \delta$*,*

$$\tilde{p}\left(max(S), max\Big(\text{KNOCKOUT-ROUND}(S, \varepsilon, \delta)\Big)\right) \leq \gamma\varepsilon.$$

KNOCKOUT

Now we present the main algorithm KNOCKOUT. KNOCKOUT takes an input set S and runs $\log n$ rounds of KNOCKOUT-ROUND halving the size of S at the end of each round. Recall that KNOCKOUT-ROUND makes sure that maximum element in the output set is comparable to maximum element in the input set. Using this, KNOCKOUT makes sure that the output element is comparable to maximum element in the input set.

Since the size of S gets halved after each round, KNOCKOUT compares each pair more times in the latter rounds. Hence the bias between maximum element in input set and maximum element in output set is small in latter rounds.

Algorithm 3 KNOCKOUT

Input: Set S, bias ε, confidence δ, stochasticity γ.
Initialize: $i = 1$, $S =$ set of all elements, $c = 2^{1/3} - 1$.
while $|S| > 1$

1. S = KNOCKOUT-ROUND$\Big(S, \frac{c\varepsilon}{\gamma^2 2^{i/3}}, \frac{\delta}{2^i}\Big)$.
2. $i = i + 1$.

Output: the unique element in S.

Note that KNOCKOUT uses only memory of set S and hence $O(n)$ memory suffices.

Theorem 4 shows that KNOCKOUT outputs an ε-maximum with probability $\geq 1 - \delta$. It also bounds the number of comparisons used by the algorithm.

Theorem 4. KNOCKOUT(S,ε,δ) *uses* $O\Big(\frac{\gamma^A|S|}{\varepsilon^2}\left(1+\log\frac{1}{\delta}\right)\Big)$ *comparisons and with probability at least* $1-\delta$*, outputs an* ε*-maximum.*

2.4 Ranking

We propose a ranking algorithm that with probability at least $1-\frac{1}{n}$ uses $O\Big(\frac{n\log n(\log\log n)^3}{\varepsilon^2}\Big)$ comparisons and outputs an ε-ranking.

Notice that we use only $\tilde{O}\Big(\frac{n\log n}{\varepsilon^2}\Big)$ comparisons for $\delta=\frac{1}{n}$ where as [SBFPH15] uses $O\big(n(\log n)^2/\varepsilon^2\big)$ comparisons even for constant error probability δ. Furthermore [SBFPH15] provided these guarantees only under Plackett-Luce model which is more restrictive compared to ours. Also, their algorithm uses $O(n^2)$ memory compared to $O(n)$ memory requirement of ours.

Our main algorithm BINARY-SEARCH-RANKING assumes the existence of a ranking algorithm RANK-x that with probability at least $1-\delta$ uses $O\Big(\frac{n}{\varepsilon^2}(\log n)^x\log\frac{n}{\delta}\Big)$ comparisons and outputs an ε-ranking for any $\delta>0$, $\varepsilon>0$ and some $x>1$. We also present a RANK-x algorithm with $x=3$.

Observe that we need RANK-x algorithm to work for any model that satisfies SST and STI. [SBFPH15] showed that their algorithm works for Plackett-Luce model but not for more general model. So we present a RANK-x algorithm that works for general model.

The main algorithm BINARY-SEARCH-RANKING randomly selects $\frac{n}{(\log n)^x}$ elements (anchors) and rank them using RANK-x . The algorithm has then effectively created $\frac{n}{(\log n)^x}$ bins, each between two successively ranked anchors. Then for each element, the algorithm identifies the bin it belongs to using a noisy binary search algorithm. The algorithm then ranks the elements within each bin using RANK-x .

We first present MERGE-RANK, a RANK-3 algorithm.

2.4.1 Merge Ranking

We present a simple ranking algorithm MERGE-RANK that uses $O\left(\frac{n(\log n)^3}{\varepsilon^2}\log\frac{n}{\delta}\right)$ comparisons, $O(n)$ memory and with probability $\geq 1-\delta$ outputs an ε-ranking. Thus MERGE-RANK is a RANK-x algorithm for $x = 3$.

Similar to Merge Sort, MERGE-RANK divides the elements into two sets of equal size, ranks them separately and combines the sorted sets. Due to the noisy nature of comparisons, MERGE-RANK compares two elements i, j sufficient times, so that the comparison output is correct with high probability when $|\tilde{p}(i,j)| \geq \frac{\varepsilon}{\log n}$. Put differently, MERGE-RANK is same as the typical Merge Sort, except it uses COMPARE as the comparison function. Due to lack of space, MERGE-RANK is presented in Appendix 2.A.

Let's define the error of an ordered set S as the maximum distance between two wrongly ordered items in S, namely,

$$err(S) \stackrel{\text{def}}{=} \max_{1\leq i\leq j\leq |S|} \tilde{p}(S(i),S(j)).$$

We show that when we merge two ordered sets, the error of the resulting ordered set will be at most $\frac{\varepsilon}{\log n}$ more than the maximum of errors of individual ordered sets.

Observe that MERGE-RANK is a recursive algorithm and the error of a singleton set is 0. Two singleton sets each containing a unique element from the input set merge to form a set with two elements with an error at most $\frac{2\varepsilon}{\log n}$, then two sets with two elements merge to form a set with four elements with an error of at most $\frac{3\varepsilon}{\log n}$ and henceforth. Thus the error of the output ordered set is bounded by ε.

Lemma 5 shows that MERGE-RANK can output an ε-ranking of S with probability $\geq 1-\delta$. It also bounds the number of comparisons used by the algorithm.

Lemma 5. MERGE-RANK$\left(S, \frac{\varepsilon}{\log|S|}, \frac{\delta}{|S|^2}\right)$ *takes* $O\left(\frac{|S|(\log|S|)^3}{\varepsilon^2}\log\frac{|S|}{\delta}\right)$ *comparisons and with probability* $\geq 1-\delta$*, outputs an* ε*-ranking. Hence,* MERGE-RANK *is a* RANK-3 *algorithm.*

Now we present our main ranking algorithm.

2.4.2 BINARY-SEARCH-RANKING

We first sketch the algorithm outline below. We then provide a proof outline.

Algorithm outline

Our algorithm is stated in BINARY-SEARCH-RANKING. It can be summarized in three major parts.

Creating anchors: (Steps 1 to 3) BINARY-SEARCH-RANKING first selects a set S' of $\frac{n}{(\log n)^x}$ random elements (anchors) and ranks them using RANK-x . At the end of this part, there are $\frac{n}{(\log n)^x}$ ranked anchors. Equivalently, the algorithm creates $\frac{n}{(\log n)^x} - 1$ bins, each bin between two successively ranked anchors.

Coarse ranking: (Step 4) After forming the bins, the algorithm uses a random walk on a binary search tree, to find which bin each element belongs to. INTERVAL-BINARY-SEARCH is similar to the noisy binary search algorithm in [FRPU94]. It builds a binary search tree with the bins as the leaves and it does a random walk over this tree. Due to lack of space the algorithm INTERVAL-BINARY-SEARCH is presented in Appendix 2.B but more intuition is given later in this section.

Ranking within each bin: (Step 5) For each bin, we show that the number of elements far from both anchors is bounded. The algorithm checks elements inside a bin whether they are close to any of the bin's anchors. For the elements that are close to anchors, the algorithm ranks them close to the anchor. And for the elements that are away from both anchors the algorithm ranks them using RANK-x and outputs the resulting ranking.

Algorithm 4 BINARY-SEARCH-RANKING

Input: Set S, bias ε.

Initialize: $\varepsilon' = \varepsilon/16$, $\varepsilon'' = \varepsilon/15$, and $S^o = \emptyset$. $S_j = \emptyset$, $C_j = \emptyset$ and $B_j = \emptyset$, for $1 \leq j \leq \left\lfloor \frac{n}{(\log n)^x} \right\rfloor + 2$.

1. Form a set S' with $\left\lfloor \frac{n}{(\log n)^x} \right\rfloor$ random elements from S. Remove these elements from S.
2. Rank S' using RANK-x $\left(S', \varepsilon', \frac{1}{n^6}\right)$.
3. Add dummy element a at the beginning of S' such that $p(a,e) = 0 \; \forall e \in S \bigcup S'$. Add dummy element b at the end of S' such that $p(b,e) = 1 \; \forall e \in S \bigcup S'$.
4. **for** $e \in S$:
   (a) k = INTERVAL-BINARY-SEARCH(S', e, ε'').
   (b) Insert e in S_k.
5. **for** $j = 1$ to $\left\lfloor \frac{n}{(\log n)^x} \right\rfloor + 2$:
   (a) **for** $e \in S_j$:
      i. **if** COMPARE2$(e, S'(j), 10\varepsilon''^{-2} \log n) \in \left[\frac{1}{2} - 6\varepsilon'', \frac{1}{2} + 6\varepsilon''\right]$, insert e in C_j.
      ii. **else if** COMPARE2$(e, S'(j+1), 10\varepsilon''^{-2} \log n) \in \left[\frac{1}{2} - 6\varepsilon'', \frac{1}{2} + 6\varepsilon''\right]$, then insert e in C_{j+1}.
      iii. **else** insert e in B_j.
   (b) Rank B_j using RANK-x $\left(B_j, \varepsilon'', \frac{1}{n^4}\right)$.
   (c) Append $S'(j)$, C_j, B_j in order at the end of S^o.

Output: S^o

Analysis of BINARY-SEARCH-RANKING

Creating anchors In Step 1 of the algorithm we select $n/(\log n)^x$ random elements. Since these are chosen uniformly random, they lie nearly uniformly in the set S. This intuition is formalized in the next lemma.

Lemma 6. *Consider a set S of n elements. If we select $\frac{n}{(\log n)^x}$ elements uniformly randomly from S and build an ordered set S' s.t. $\tilde{p}(S'(i), S'(j)) \geq 0 \; \forall i > j$, then with probability $\geq 1 - \frac{1}{n^4}$, for*

Algorithm 5 COMPARE2

Input: element i, element j, number of comparisons m.

1. Compare i and j for m times and return the fraction of times i wins over j.

any $\varepsilon > 0$ *and all* k,

$$|\{e \in S : \tilde{p}(e, S'(k)) > \varepsilon, \tilde{p}(S'(k+1), e) > \varepsilon\}| \leq 5(\log n)^{x+1}.$$

In Step 2, we use RANK-x to rank S'. Lemma 7 shows the guarantee of ranking S'.

Lemma 7. *After Step 2 of the* BINARY-SEARCH-RANKING *with probability* $\geq 1 - \frac{1}{n^6}$, S' *is* ε'*-ranked.*

At the end of Step 2, we have $\frac{n}{(\log n)^x} - 1$ bins, each between two successively ranked anchors. Each bin has a left anchor and a right anchor . We say that an element belongs to a bin if it wins over the bin's left anchor with probability $\geq \frac{1}{2}$ and wins over the bin's right anchor with probability $\leq \frac{1}{2}$. Notice that some elements might win over $S'(1)$ with probability $< \frac{1}{2}$ and thus not belong to any bin. So in Step 3, we add a dummy element a at the beginning of S' where a loses to every element in $S \bigcup S'$ with probability 1. For similar reasons we add a dummy element b to the end of S' where every element in $S \bigcup S'$ loses to b with probability 1.

Coarse Ranking Note that $S'(i)$ and $S'(i+1)$ are respectively the left and right anchors of the bin S_i.

Since S' is ε'-ranked and the comparisons are noisy, it is hard to find a bin S_i for an element e such that $p(e, S'(i)) \geq \frac{1}{2}$ and $p(S'(i+1), e) \geq \frac{1}{2}$. We call a bin S_i a $\varepsilon''-$*nearly correct* bin for an element e if $p(e, S'(i)) \geq \frac{1}{2} - \varepsilon''$ and $p(S'(i+1), e) \geq \frac{1}{2} - \varepsilon''$ for some $\varepsilon'' > \varepsilon'$.

In Step 4, for each element we find an ε''*-nearly correct* bin using INTERVAL-BINARY-SEARCH . Next we describe an outline of INTERVAL-BINARY-SEARCH.

INTERVAL-BINARY-SEARCH first builds a binary search tree of intervals (see Ap-

pendix 2.B) as follows: the root node is the entire interval between the first and the last elements in S'. Each non-leaf node interval I has two children corresponding to the left and right halves of I. The leaves of the tree are the bins between two successively ranked anchors.

To find an ε''-*nearly correct* bin for an element e, the algorithm starts at the root of the binary search tree and at every non-leaf node corresponding to interval I, it checks if e belongs to I or not by comparing e with I's left and right anchors. If e loses to left anchor or wins against the right anchor, the algorithm backtracks to current node's parent.

If e wins against I's left anchor and loses to its right one, the algorithm checks if e belongs to the left or right child by comparing e with the middle element of I and moves accordingly.

When at a leaf node, the algorithm checks if e belongs to the bin by maintaining a counter. If e wins against the bin's left anchor and loses to the bin's right anchor, it increases the counter by one or otherwise it decreases the counter by one. If the counter is less than 0 the algorithm backtracks to the bin's parent. By repeating each comparison several times, the algorithm makes a correct decision with probability $\geq \frac{19}{20}$.

Note that there could be several ε''-*nearly correct* bins for e and even though at each step the algorithm moves in the direction of one of them, it could end up moving in a loop and never reaching one of them. We thus run the algorithm for $30\log n$ steps and terminate.

If the algorithm is at a leaf node by $30\log n$ steps and the counter is more than $10\log n$ we show that the leaf node bin is a ε''-*nearly correct* bin for e and the algorithm outputs the leaf node. If not, the algorithm puts in a set Q all the anchors visited so far and orders Q according to S'.

We select $30\log n$ steps to ensure that if there is only one nearly correct bin, then the algorithm outputs that bin w.p. $\geq 1 - \frac{1}{n^6}$. Also we do not want too many steps so as to bound the size of Q.

By doing a simple binary search in Q using BINARY-SEARCH (see Appendix 2.B) we find an anchor $f \in Q$ such that $|\tilde{p}(e,f)| \leq 4\varepsilon''$. Since INTERVAL-BINARY-SEARCH ran for at most $30\log n$ steps, Q can have at most $60\log n$ elements and hence BINARY-SEARCH can search

effectively by repeating each comparison $O(\log n)$ times to maintain high confidence. Next paragraph explains how BINARY-SEARCH finds such an element f.

BINARY-SEARCH first compares e with the middle element m of Q for $O(\log n)$ times. If the fraction of wins for e is between $\frac{1}{2} - 3\varepsilon''$ and $\frac{1}{2} + 3\varepsilon''$, then w.h.p. $|\tilde{p}(e,m)| \leq 4\varepsilon''$ and hence BINARY-SEARCH outputs m. If the fraction of wins for e is less than $\frac{1}{2} - 3\varepsilon''$, then w.h.p. $\tilde{p}(e,m) \leq -2\varepsilon''$ and hence it eliminates all elements to the right of m in Q. If the fraction of wins for e is more than $\frac{1}{2} + 3\varepsilon''$, then w.h.p. $\tilde{p}(e,m) \geq 2\varepsilon''$ and hence it eliminates all elements to the left of m in Q. It continues this process until it finds an element f such that the fraction of wins for e is between $\frac{1}{2} - 3\varepsilon''$ and $\frac{1}{2} + 3\varepsilon''$.

In next Lemma, we show that INTERVAL-BINARY-SEARCH achieves to find a $5\varepsilon''$-*nearly correct* bin for every element.

Lemma 8. *For any element $e \in S$, Step 4 of* BINARY-SEARCH-RANKING *places e in bin S_l such that $\tilde{p}(e,S'(l)) > -5\varepsilon''$ and $\tilde{p}(S'(l+1),e) > -5\varepsilon''$ with probability $\geq 1 - \frac{1}{n^5}$.*

Ranking within each bin Once we have identified the bins, we rank the elements inside each bin. By Lemma 6, inside each bin all elements are close to the bin's anchors except at most $5(\log n)^{x+1}$ of them.

The algorithm finds the elements close to anchors in Step 5*a* by comparing each element in the bin with the bin's anchors. If an element in bin S_j is close to bin's anchors $S'(j)$ or $S'(j+1)$, the algorithm moves it to the set C_j or C_{j+1} accordingly and if it is far away from both, the algorithm moves it to the set B_j. The following two lemmas state that this separating process happens accurately with high probability. The proofs of these results follow from the Chernoff bound and hence omitted.

Lemma 9. *At the end of Step 5a, for all j, $\forall e \in C_j$, $|\tilde{p}(e,S'(j))| < 7\varepsilon''$ with probability $\geq 1 - \frac{1}{n^3}$.*

Lemma 10. *At the end of Step 5a, for all j, $\forall e \in B_j$, $\min(\tilde{p}(e,S'(j)), \tilde{p}(S'(j+1),e)) > 5\varepsilon''$ with probability $\geq 1 - \frac{1}{n^3}$.*

Combining Lemmas 6, 7 and 10 next lemma shows that the size of B_j is bounded for all j.

Lemma 11. *At the end of Step 5a, $|B_j| \leq 5(\log n)^{x+1}$ for all j, with probability $\geq 1 - \frac{3}{n^3}$.*

Since all the elements in C_j are already close to an anchor, they need not be ranked. By Lemma 11 with probability $\geq 1 - \frac{3}{n^3}$ the number of elements in B_j is at most $5(\log n)^{x+1}$. We use RANK-x to rank each B_j and output the final ranking.

Lemma 12 shows that all B_j's are ε''-ranked at the end of Step 5b. Proof follows from properties of RANK-x and union bound.

Lemma 12. *At the end of Step 5b, all B_js are ε''-ranked with probability $\geq 1 - \frac{1}{n^3}$.*

Combining the above set of results yields our main result.

Theorem 13. *Given access to* RANK-x, BINARY-SEARCH-RANKING *with probability $\geq 1 - \frac{1}{n}$, uses $O\left(\frac{n \log n (\log \log n)^x}{\varepsilon^2}\right)$ comparisons and outputs an ε-ranking.*

Using MERGE-RANK as a RANK-x algorithm with $x = 3$ leads to the following corollary.

Corollary 14. BINARY-SEARCH-RANKING *uses $O\left(\frac{n \log n (\log \log n)^3}{\varepsilon^2}\right)$ comparisons and outputs an ε-ranking with probability $\geq 1 - \frac{1}{n}$.*

Using PALPAC-AMPRR [SBFPH15] as a RANK-x algorithm with $x = 1$ leads to the following corollary over PL model.

Corollary 15. *Over PL model,* BINARY-SEARCH-RANKING *with probability $\geq 1 - \frac{1}{n}$ uses $O\left(\frac{n \log n \log \log n}{\varepsilon^2}\right)$ comparisons and outputs an ε-ranking.*

It is well known that to rank a set of n values under the noiseless setting, $\Omega(n \log n)$ comparisons are necessary. We show that under the noisy model, $\Omega\left(\frac{n}{\varepsilon^2} \log \frac{n}{\delta}\right)$ samples are necessary to output an ε-ranking and hence our algorithm is near-optimal.

Theorem 16. *For $\varepsilon \leq \frac{1}{4}$, $\delta \leq \frac{1}{2}$, there exists a noisy model that satisfies SST and STI such that to output an ε-ranking with probability $\geq 1 - \delta$, $\Omega\left(\frac{n}{\varepsilon^2} \log \frac{n}{\delta}\right)$ comparisons are necessary.*

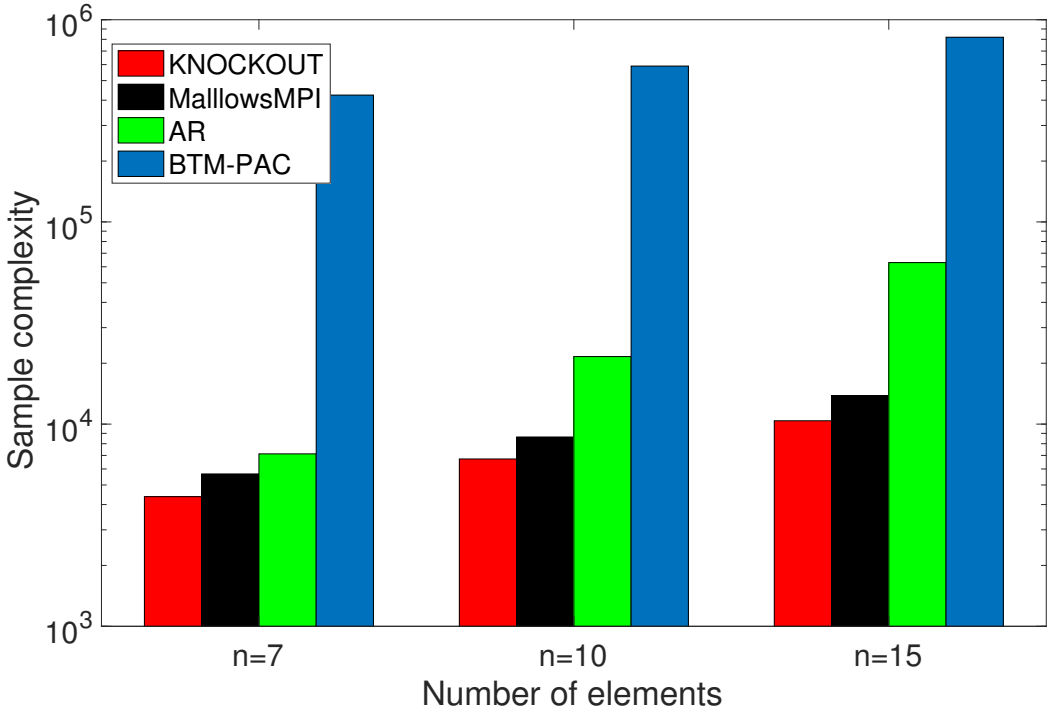


Figure 2.1: Comparison of sample complexity for small input sizes, with $\varepsilon = 0.05$, and $\delta = 0.1$

2.5 Experiments

We compare the performance of our algorithms with that of others over simulated data. Similar to [YJ11], we consider the stochastic model where $p(i, j) = 0.6 \; \forall i < j$. Note that this model satisfies both SST and STI. We find 0.05-maximum with error probability $\delta = 0.1$. Observe that $i = 1$ is the only 0.05-maximum. We compare the sample complexity of KNOCKOUT with that of **BTM-PAC** [YJ11], **MallowsMPI** [BFHS14], and **AR** [HSRW16]. **BTM-PAC** is an (ε, δ)-PAC algorithm for the same model considered in this paper. **MallowsMPI** finds a Condorcet winner which exists under our general model. **AR** finds the maximum according to *Borda* scores. We also tried **PLPAC** [SBFPH15], developed originally for PL model but the algorithm could not meet guarantees of $\delta = 0.1$ under this model and hence omitted. Note that in all the experiments the reported numbers are averaged over 100 runs.

In Figure 2.1, we compare the sample complexity of algorithms when there are 7, 10

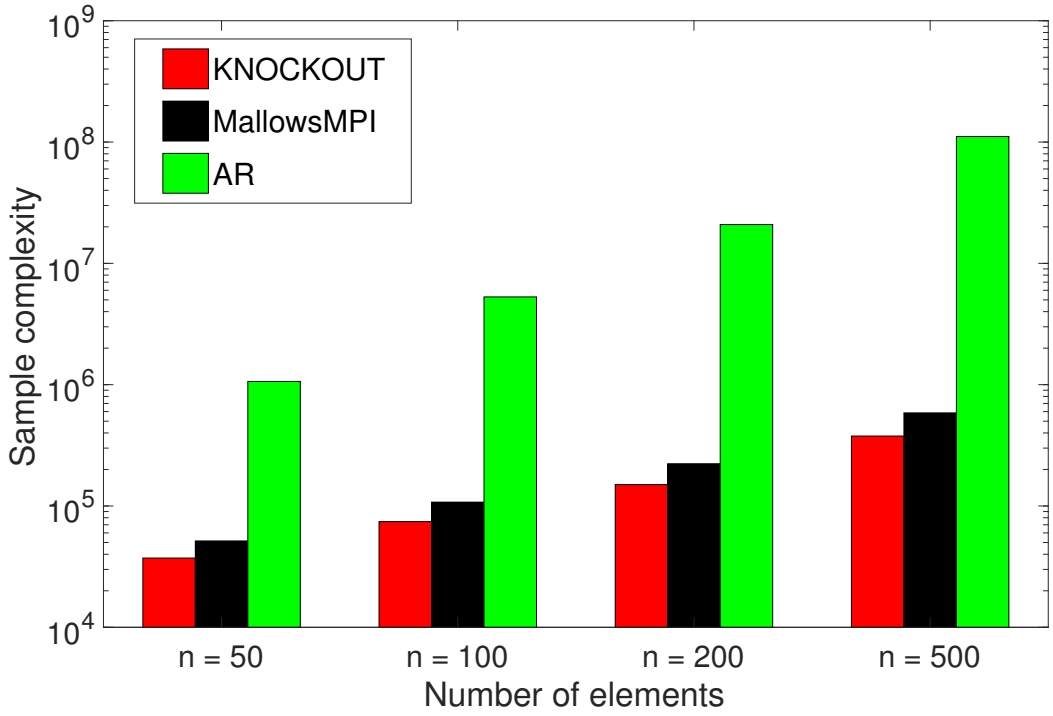


Figure 2.2: Comparison of sample complexity for large input size, with $\varepsilon = 0.05$, and $\delta = 0.1$

and 15 elements. Our algorithm outperforms all the others. **BTM-PAC** performs much worse in comparison to others because of high constants in the algorithm. Further **BTM-PAC** allows comparing an element with itself since the main objective in [YJ11] is to reduce the regret. We exclude **BTM-PAC** for further experiments with higher number of elements.

In Figure 2.2, we compare the algorithms when there are 50, 100, 200 and 500 elements. Our algorithm outperforms others for higher number of elements too. Performance of **AR** gets worse as the number of elements increases since Borda scores of the elements get closer to each other and hence **AR** takes more comparisons to eliminate an element. Notice that number of comparisons is in logarithmic scale and hence the performance of **MallowsMPI** appears to be close to that of ours.

As noted in [SBFPH15], sample complexity of **MallowsMPI** gets worse as $\tilde{p}(i,j)$ gets close to 0. To show the pronounced effect, we use the stochastic model $p(1,j) = 0.6\ \forall j > 1$, $p(i,j) = 0.5 + \tilde{q}\ \forall j > i, i > 1$ where $\tilde{q} < 0.1$, and the number of elements is 15. Here too we find

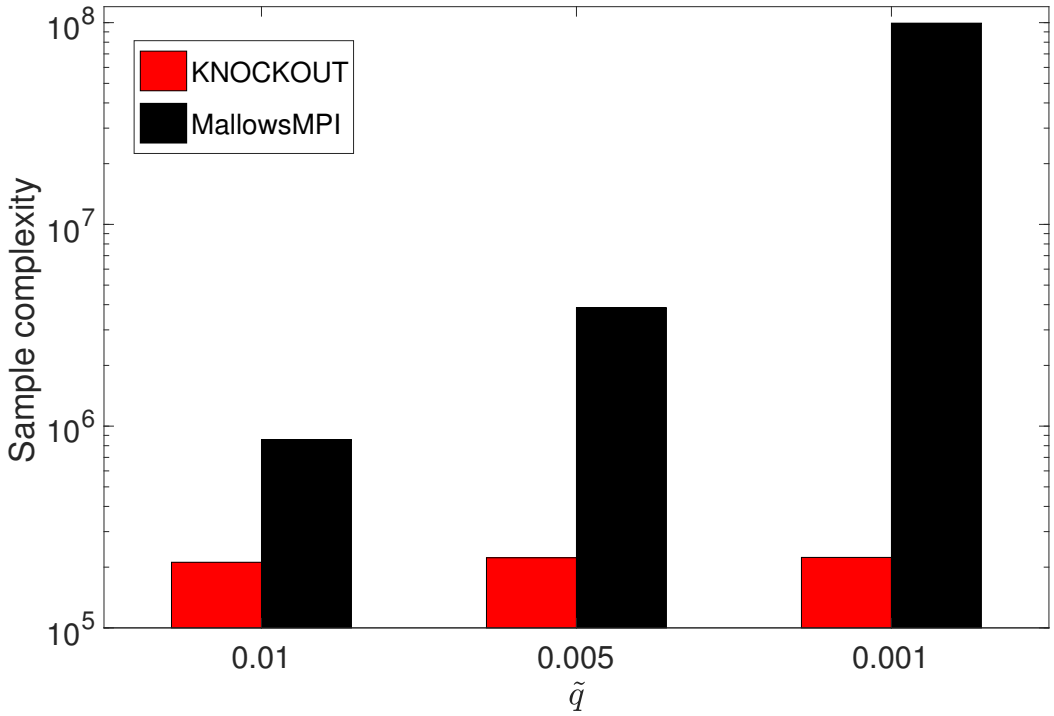


Figure 2.3: Sample complexity of KNOCKOUT and MallowsMPI for different values of $\tilde{q}$, with $\varepsilon = 0.05$ and $\delta = 0.1$

0.05-maximum with $\delta = 0.1$. Note that $i = 1$ is the only 0.05-maximum in this stochastic model. In Figure 2.3, we compare the algorithms for different values of $\tilde{q}$: 0.01, 0.005 and 0.001. As discussed above, the performance of **MallowsMPI** gets much worse whereas our algorithm's performance stays unchanged. The reason is that **MallowsMPI** finds the Condorcet winner using successive elimination technique and as $\tilde{q}$ gets closer to 0, **MallowsMPI** takes more comparisons for each elimination. Our algorithm tries to find an alternative which defeats Condorcet winner with probability $\geq 0.5 - 0.05$ and hence for alternatives that are very close to each other, our algorithm declares either one of them as winner after comparing them for certain number of times.

Next we evaluate KNOCKOUT on Mallows model which does not satisfy STI. Mallows is a parametric model which is specified by single parameter ϕ. As in [BFHS14], we consider $n = 10$ elements and various values for ϕ: 0.03, 0.1, 0.3, 0.5, 0.7, 0.8, 0.9, 0.95 and 0.99. Here again

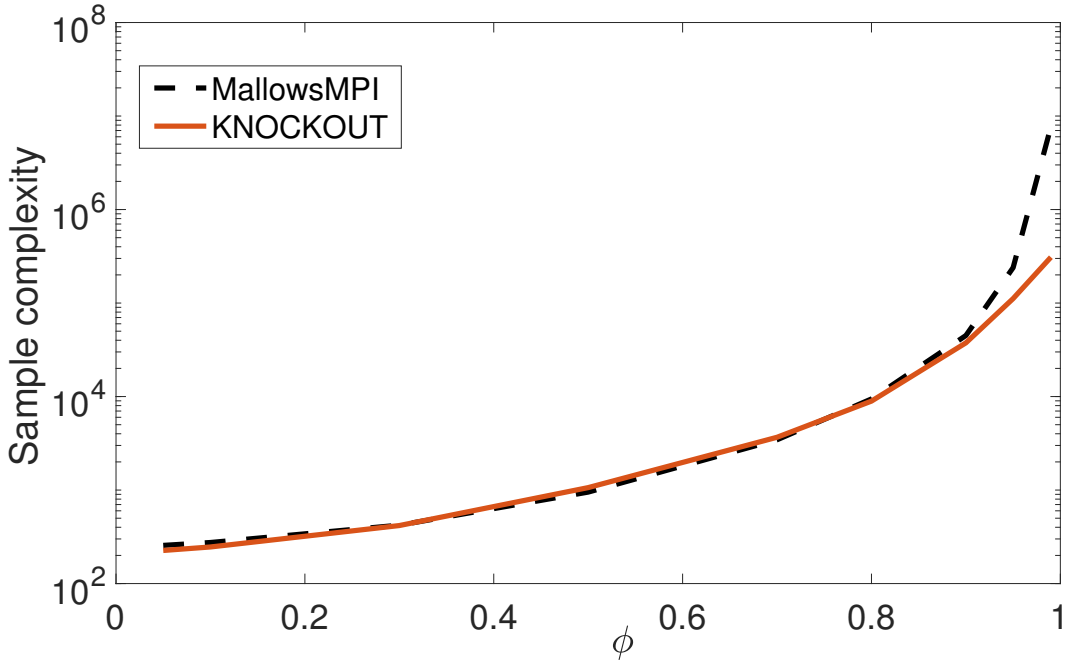


Figure 2.4: Sample complexity of KNOCKOUT and **MallowsMPI** under Mallows model for various values of ϕ

we seek to find 0.05-maximum with $\delta = 0.05$. As we can see in Figure 2.4, sample complexity of KNOCKOUT and **MallowsMPI** is essentially same under small values of ϕ but KNOCKOUT outperforms **MallowsMPI** as ϕ gets close to 1 since comparison probabilities grow closer to $1/2$. Surprisingly, for all values of ϕ except for 0.99, KNOCKOUT returned Condorcet winner in all runs. For $\phi = 0.99$, KNOCKOUT returned second best element in 10 runs out of 100. Note that $\tilde{p}(1,2) = 0.0025$ and hence KNOCKOUT still outputed a 0.05-maximum. Even though we could not show theoretical guarantees of KNOCKOUT under Mallows model, our simulations suggest that it can perform well even under this model.

For the stochastic model $p(i,j) = 0.6\ \forall i < j$, we run our MERGE-RANK algorithm to find an ε-ranking with $\delta = 0.1$. Figure 2.5 shows that sample complexity does not increase a lot with decreasing ε. We attribute this to the subroutine COMPARE that finds the winner faster when the elements are more dissimilar.

Some more experiments are provided in Appendix 2.G.

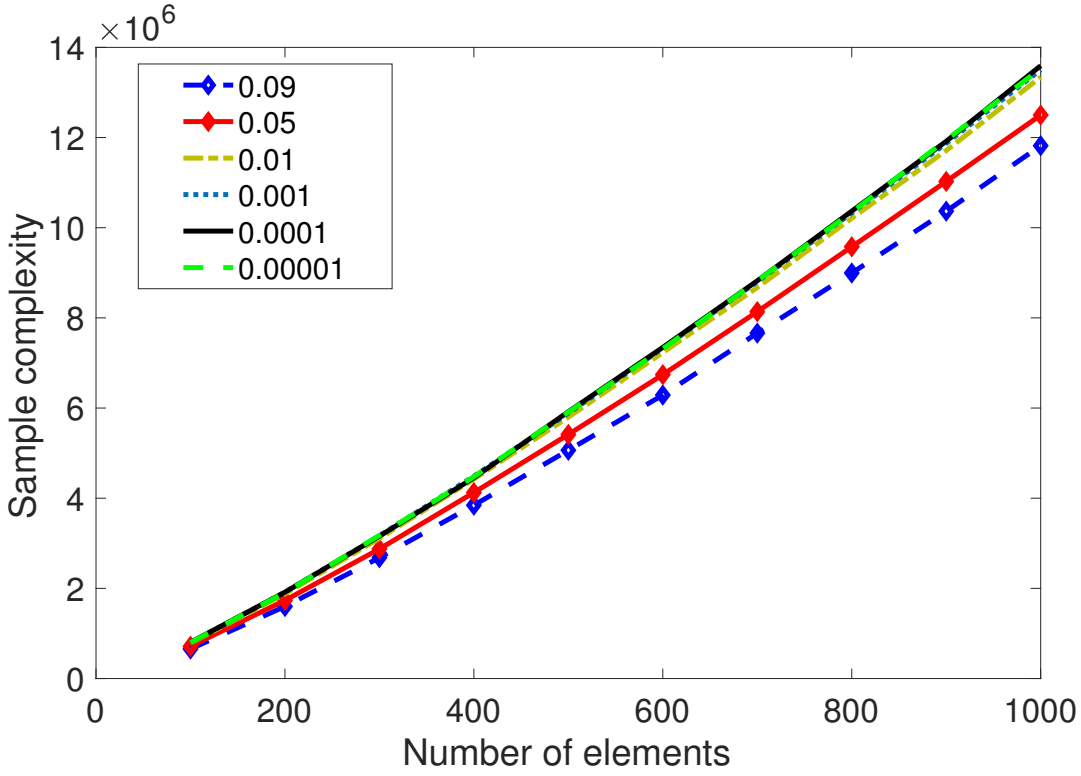


Figure 2.5: Sample complexity of MERGE-RANK for different ε

2.6 Conclusion

We studied maximum selection and ranking using noisy comparisons for broad comparison models satisfying SST and STI. For maximum selection we presented a simple algorithm with linear, hence optimal, sample complexity. For ranking we presented a framework that improves the performance of many ranking algorithms and applied it to merge ranking to derive a near-optimal algorithm.

We conducted several experiments showing that our algorithms perform well and outperform existing algorithms on simulated data.

The maximum-selection experiments suggest that our algorithm performs well even without STI. It would be of interest to extend our theoretical guarantees to this case. For ranking, it would be interesting to close the $(\log\log n)^3$ ratio between the upper- and lower- complexity bounds.

2.7 Acknowledgement

Chapter 2, in full, is a reprint of the material as it appears in *International Conference on Machine Learning*. Falahatgar, Moein, Alon Orlitsky, Venkatadheeraj Pichapati, and Ananda Theertha Suresh, 2017. The dissertation author was the primary investigator and author of this paper.

2.A Merge Ranking

We first introduce a subroutine that is used by MERGE-RANK. It merges two ordered sets in the presence of noisy comparisons.

2.A.1 MERGE

MERGE takes two ordered sets S_1 and S_2 and outputs an ordered set Q by merging them. MERGE starts by comparing the first elements in each set S_1 and S_2 and places the loser in the first position of Q. It compares the two elements sufficient times to make sure that output is near-accurate. Then it compares the winner and the element right to loser in the corresponding set. It continues this process until we run out of one of the sets and then adds the remaining elements to the end of Q and outputs Q.

We show that when we merge two ordered sets using MERGE, the error of resulting ordered set is not high compared to the maximum of errors of individual ordered sets.

Lemma 17. *With probability* $\geq 1-(|S_1|+|S_2|)\delta$*, error of* $\text{MERGE}(S_1,S_2,\varepsilon,\delta)$ *is at most* ε *more than the maximum of errors of* S_1 *and* S_2*. Namely, with probability* $\geq 1-(|S_1|+|S_2|)\delta$*,*

$$err(\text{MERGE}(S_1,S_2,\varepsilon,\delta)) \leq \max\left(err(S_1),err(S_2)\right)+\varepsilon.$$

Algorithm 6 MERGE

Input: Sets S_1, S_2, bias ε, confidence δ.
Initialize: $i = 1$, $j = 1$ and $O = \emptyset$.

1. **while** $i \leq |S_1|$ and $j \leq |S_2|$.
   (a) **if** $S_1(i) = \text{COMPARE}(S_1(i), S_2(j), \varepsilon, \delta)$, then append $S_2(j)$ at the end of O and $j = j+1$.
   (b) **else** append $S_1(i)$ at the end of O and $i = i+1$.
2. **if** $i \leq |S_1|$, then append $S_1(i : |S_1|)$ at the end of O.
3. **if** $j \leq |S_2|$, then append $S_2(j : |S_2|)$ at the end of O.

Output: O.

2.A.2 MERGE-RANK

Now we present the algorithm MERGE-RANK. MERGE-RANK partitions the input set S into two sets S_1 and S_2 each of size $|S|/2$. It then orders S_1 and S_2 separately using MERGE-RANK and combines the ordered sets using MERGE. Notice that MERGE-RANK is a recursive algorithm. The singleton sets each containing an unique element in S are merged first. Two singleton sets are merged to form a set with two elements, then the sets with two elements are merged to form a set with four elements and henceforth. By Lemma 17, each merge with bound parameter ε' adds at most ε' to the error. Since error of singleton sets is 0 and each element takes part in $\log n$ merges, the error of the output set is at most $\varepsilon' \log n$. Hence with bound parameter $\varepsilon/\log n$, the error of the output set is less than ε.

Algorithm 7 MERGE-RANK

Input: Set S, bias ε, confidence δ.

1. $S_1 = \text{MERGE-RANK}(S(1 : \lfloor |S|/2 \rfloor), \varepsilon, \delta)$.
2. $S_2 = \text{MERGE-RANK}(S(\lfloor |S|/2 \rfloor + 1 : |S|), \varepsilon, \delta)$.

Output: $\text{MERGE}(S_1, S_2, \varepsilon, \delta)$.

2.B Algorithms for Ranking

Algorithm 8 INTERVAL-BINARY-SEARCH

Input: Ordered array S, search element e, bias ε

1. T = BUILD-BINARY-SEARCH-TREE($|S|$).
2. Initialize set $Q = \emptyset$, node $\alpha = \text{root}(T)$, and count $c = 0$.
3. **repeat** for $30\log n$ times
   (a) **if** $\alpha_2 - \alpha_1 > 1$,
      i. Add α_1, α_2 and $\left\lceil \frac{\alpha_1+\alpha_2}{2} \right\rceil$ to Q.
      ii. **if** COMPARE2$(S(\alpha_1), e, \frac{10}{\varepsilon^2}) > 1/2$ or COMPARE2$(e, S(\alpha_2), \frac{10}{\varepsilon^2}) > 1/2$ then go back to the parent, $\alpha = \text{parent}(\alpha)$.
      iii. **else**
         - **if** COMPARE2$(S(\left\lceil \frac{\alpha_1+\alpha_2}{2} \right\rceil), e, \frac{10}{\varepsilon^2}) > 1/2$ go to the left child,$\alpha = \text{left}(\alpha)$.
         - **else** go to the right child, $\alpha = \text{right}(\alpha)$.

   (b) **else**
      i. **if** COMPARE2$(e, S(\alpha_1), \frac{10}{\varepsilon^2}) > \frac{1}{2}$ and COMPARE2$(S(\alpha_2), e, \frac{10}{\varepsilon^2}) > \frac{1}{2}$, $c = c + 1$.
      ii. **else**
         A. **if** $c = 0$, $\alpha = \text{parent}(\alpha)$.
         B. **else** $c = c - 1$.
4. (a) **if** $c > 10\log n$, **Output:** α_1.
   (b) **else**
      i. Sort Q.
      ii. **Output:** BINARY-SEARCH(S, Q, e, ε).

Algorithm 9 BUILD-BINARY -SEARCH-TREE

Input: size n.

// Recall that each node m in the tree is an interval between left end m_1 and right end m_2.

1. Initialize set $T' = \emptyset$.

2. Initialize the tree T with the root node $(1,n)$.

$$m = (1,n) \qquad \text{where } m_1 = 1 \text{ and } m_2 = n,$$
$$\text{root}(T) = m$$

3. Add m to T'.

4. **while** T' is not empty

   (a) Consider a node i in T'.

   (b) **if** $i_2 - i_1 > 1$, create a left child and right child to i and set their parents as i.

$$\alpha = \left(i_1, \left\lceil \frac{i_1 + i_2}{2} \right\rceil\right), \qquad \beta = \left(\left\lceil \frac{i_1 + i_2}{2} \right\rceil, i_2\right),$$
$$\text{left}(i) = \alpha, \qquad \text{right}(i) = \beta,$$
$$\text{parent}(\alpha) = i, \qquad \text{parent}(\beta) = i.$$

   and add nodes α and β to T'.

   (c) Remove node i from T'.

Output: T.

Algorithm 10 BINARY-SEARCH

Input: Ordered array S, ordered array Q, search item e, bias ε.

Initialize: $l = 1$, $h = |Q|$.

1. **while** $h - l > 0$

   (a) $t = \text{COMPARE2}\left(e, S(Q(\lceil \frac{l+h}{2} \rceil), \frac{10 \log n}{\varepsilon^2}\right)$.

   (b) **if** $t \in \left[\frac{1}{2} - 3\varepsilon, \frac{1}{2} + 3\varepsilon\right]$, then **Output:** $Q(\lceil \frac{l+h}{2} \rceil)$.

   (c) **else if** $t < \frac{1}{2} - 3\varepsilon$, then move to the left.

$$h = \left\lceil \frac{l+h}{2} \right\rceil.$$

   (d) **else** move to the right.

$$l = \left\lceil \frac{l+h}{2} \right\rceil.$$

Output: $Q(h)$.

2.C Some tools for proving lemmas

We first prove an auxilliary result that we use in the future analysis.

Lemma 18. *Let* $W = \text{COMPARE}(i, j, \varepsilon, \delta)$ *and* L *be the other element. Then with probability* $\geq 1 - \delta$,

$$p(W, L) \geq \frac{1}{2} - \varepsilon.$$

Proof. Note that if $|\tilde{p}(i, j)| < \varepsilon$, then $p(i, j) > \frac{1}{2} - \varepsilon$ and $p(j, i) > \frac{1}{2} - \varepsilon$. Hence, $p(W, L) \geq \frac{1}{2} - \varepsilon$.

If $|\tilde{p}(i, j)| \geq \varepsilon$, without loss of generality, assume that i is a better element i.e., $\tilde{p}(i, j) \geq \varepsilon$.

By Lemma 2, with probability atleast $1-\delta$, $W = i$. Hence

$$Pr\left(p(W,L) \geq \frac{1}{2} - \varepsilon\right) = Pr(W = i) \geq 1 - \delta. \qquad \square$$

We now prove a Lemma that follows from SST and STI that we will use in future analysis.

Lemma 19. *If $\tilde{p}(i,j) \leq \varepsilon_1$, $\tilde{p}(j,k) \leq \varepsilon_2$, then $\tilde{p}(i,k) \leq \varepsilon_1 + \varepsilon_2$.*

Proof. We will divide the proof into four cases based on whether $\tilde{p}(i,j) > 0$ and $\tilde{p}(j,k) > 0$.

If $\tilde{p}(i,j) \leq 0$ and $\tilde{p}(j,k) \leq 0$, then by SST, $\tilde{p}(i,k) \leq 0 \leq \varepsilon_1 + \varepsilon_2$.

If $0 < \tilde{p}(i,j) \leq \varepsilon_1$ and $0 < \tilde{p}(j,k) \leq \varepsilon_2$, then by STI, $\tilde{p}(i,k) \leq \varepsilon_1 + \varepsilon_2$.

If $\tilde{p}(i,j) < 0$ and $0 < \tilde{p}(j,k) \leq \varepsilon_2$, then by SST, $\tilde{p}(i,k) \leq \varepsilon_2 \leq \varepsilon_1 + \varepsilon_2$.

If $0 < \tilde{p}(i,j) \leq \varepsilon_1$ and $\tilde{p}(j,k) < 0$, then by SST, $\tilde{p}(i,k) \leq \varepsilon_1 \leq \varepsilon_1 + \varepsilon_2$. $\square$

2.D Proofs of Section 3.3

Proof of Lemma 2

Proof. Let $\hat{p}_i^r$ and $\hat{c}^r$ denote $\hat{p}_i$ and $\hat{c}$ respectively after r number of comparisons. Output of COMPARE(i,j,ε,δ) will not be i only if $\hat{p}_i^r < \frac{1}{2} + \varepsilon - \hat{c}^r$ for any $r < m = \frac{1}{2\varepsilon^2}\log\frac{2}{\delta}$ or if $\hat{p}_i < \frac{1}{2}$ for $r = m$. We will show that the probability of each of these events happening is bounded by $\frac{\delta}{2}$. Hence by union bound, Lemma follows.

After r comparisons, by Hoeffding's inequality,

$$Pr(\hat{p}_i^r < \frac{1}{2} + \varepsilon - \hat{c}^r) \leq e^{-2r(\hat{c}^r)^2} = e^{-\log\frac{4r^2}{\delta}} = \frac{\delta}{4r^2}.$$

Using union bound,

$$Pr(\exists r \text{ s.t. } \hat{p}_i^r \leq \frac{1}{2} + \varepsilon - \hat{c}^r) \leq \frac{\delta}{2}$$

After $m = \frac{1}{2\varepsilon^2}\log\frac{2}{\delta}$ rounds, by Hoeffding's inequality,

$$Pr(\hat{p}_i^m < \frac{1}{2}) \leq e^{-2m\varepsilon^2} = \frac{\delta}{2}.$$ □

Proof of Lemma 3

Proof. Each of the $\frac{|S|}{2}$ pairs is compared at most $\frac{1}{2\varepsilon^2}\log\frac{2}{\delta}$ times, hence the total comparisons is $\leq \frac{|S|}{4\varepsilon^2}\log\frac{2}{\delta}$. Let $k^* = \max(\textsc{Knockout-Round}(S,\varepsilon,\delta))$ and $s^* = \max(S)$. Let a be the element paired with s^*. There are two cases: $\tilde{p}(s^*,a) \geq \varepsilon$ and $\tilde{p}(s^*,a) < \varepsilon$.

If $\tilde{p}(s^*,a) \geq \varepsilon$, by Lemma 2 with probability $\geq 1-\delta$, s^* will win and hence by definitions of s^* and k^*, $\tilde{p}(s^*,k^*) = 0 \leq \gamma\varepsilon$. Alternatively, if $\tilde{p}(s^*,a) < \varepsilon$, let $\text{winner}(s^*,a)$ denote the winner between s^* and a. Then,

$$r(a) \overset{(a)}{\leq} r(\text{winner}(s^*,a)) \overset{(b)}{\leq} r(k^*) \overset{(c)}{\leq} r(s^*)$$

where (a) follows from $r(a) \leq r(s^*)$, (b) and (c) follow from the definitions of s^* and k^* respectively. From stochastic tranisitivity on a, k^* and s^*, $\tilde{p}(s^*,k^*) \leq \gamma\tilde{p}(s^*,a) \leq \gamma\varepsilon$. □

Proof of Theorem 4

Proof. We first bound the number of comparisons. Let $n_i = \frac{|S|}{2^{i-1}}$ be the number of elements in the set at the beginning of round i. The number of comparisons at round i is

$$\leq \frac{n_i}{2} \cdot \frac{\gamma^4 2^{2i/3}}{2c^2\varepsilon^2} \cdot \log\frac{2^{i+1}}{\delta}.$$

Hence the number of comparisons in all rounds is

$$\sum_{i=1}^{\log|S|} \frac{|S|}{2^i} \cdot \frac{\gamma^4 2^{2i/3}}{2c^2\varepsilon^2} \cdot \log\frac{2^{i+1}}{\delta} \leq \frac{|S|\gamma^4}{2c^2\varepsilon^2} \sum_{i=1}^{\infty} \frac{1}{2^{i/3}} \left(i + \log\frac{2}{\delta}\right)$$
$$= \frac{|S|\gamma^4}{2c^2\varepsilon^2} \left(\frac{2^{1/3}}{c^2} + \frac{1}{c}\log\frac{2}{\delta}\right)$$
$$= O\left(\frac{|S|\gamma^4}{\varepsilon^2}\left(1 + \log\frac{1}{\delta}\right)\right).$$

We now show that with probability $\geq 1-\delta$, the output of KNOCKOUT is an ε-maximum. Let $\varepsilon_i = c\varepsilon/(\gamma 2^{i/3})$ and $\delta_i = \delta/2^i$. Note that ε_i and δ_i are bias and confidence values used in round i. Let b_i be a maximum element in the set S before round i. Then by Lemma 3, with probability $\geq 1-\delta_i$,

$$\tilde{p}(b_i, b_{i+1}) \leq \varepsilon_i \tag{2.1}$$

By union bound, the probability that Equation 2.1 does not hold for some round $1 \leq i \leq \log|S|$ is

$$\leq \sum_{i=1}^{\log|S|} \delta_i = \sum_{i=1}^{\log|S|} \frac{\delta}{2^i} \leq \delta.$$

With probability $\geq 1-\delta$, Equation 2.1 holds for all i and by stochastic triangle inequality,

$$\tilde{p}(b_1, b_{\log|S|+1}) \leq \sum_{i=1}^{\log|S|} \tilde{p}(b_i, b_{i+1}) \leq \sum_{i=1}^{\infty} \frac{c\varepsilon}{\gamma 2^{i/3}} = \varepsilon/\gamma.$$

We now show that if $\tilde{p}(b_1, e) \leq \varepsilon/\gamma$, e is an ε-maximum, namely $\tilde{p}(f, e) \leq \varepsilon \ \forall f \in S$. Note that b_1 is a maximum element in the original set S and hence $r(b_1) = n$. If $r(f) \geq r(e)$, then by γ-stochastic transitivity, $\tilde{p}(f, e) \leq \gamma\tilde{p}(b_1, e) \leq \varepsilon$ and if $r(f) \leq r(e)$, then $\tilde{p}(f, e) \leq 0 \leq \varepsilon$. □

2.E Proofs of Section 2.4.1

Proof of Lemma 17

Proof. Let $Q = \text{MERGE}(S_1, S_2, \varepsilon, \delta)$. We will show that for every k, w.p. $\geq 1-\delta$, $\tilde{p}(Q(k), Q(l)) \leq \max(err(S_1), err(S_2)) + \varepsilon \ \forall l > k$. Note that if this property is true for every element then $err(Q) \leq \max(err(S_1), err(S_2)) + \varepsilon$. Since there are $|S_1| + |S_2|$ elements in the final merged set, the Lemma follows by union bound.

If $S_1(i)$ and $S_2(j)$ are compared in MERGE algorithm, without loss of generality, assume that $S_1(i)$ loses i.e., $S_1(i)$ appears before $S_2(j)$ in T. The elements that appear to the right of $S_1(i)$ in Q belong to set $Q_{\geq S_1(i)} = \{S_1(k) : k > i\} \bigcup \{S_2(k) : k \geq j\}$. We will show that w.p. $\geq 1-\delta$, $\forall e \in Q_{\geq S_1(i)}$, $\tilde{p}(S_1(i), e) \leq \max(err(S_1), err(S_2)) + \varepsilon$.

By definition of error of an ordered set,

$$\tilde{p}(S_1(i), S_1(k)) \leq err(S_1) \quad \forall k > i \tag{2.2}$$

$$\tilde{p}(S_2(j), S_2(k)) \leq err(S_2) \quad \forall k \geq j. \tag{2.3}$$

By Lemma 18, w.p. $\geq 1-\delta$,

$$\tilde{p}(S_1(i), S_2(j)) \leq \varepsilon. \tag{2.4}$$

Hence by Equations 2.3, 2.4 and Lemma 19, w.p. $\geq 1-\delta$, $\tilde{p}(S_1(i), S_2(k)) \leq \varepsilon + err(S_2) \ \forall k \geq j$. □

Proof of Lemma 5

Proof. We first bound the total comparisons. Let $C(Q,\varepsilon',\delta')$ be the number of comparisons that the MERGE-RANK uses on a set Q. Since MERGE-RANK is a recursive algorithm,

$$\begin{aligned} C(Q,\varepsilon',\delta') \leq & C(Q[1:\lfloor |Q|/2 \rfloor],\varepsilon',\delta') \\ & + C(Q[\lfloor |Q|/2 \rfloor : |Q|],\varepsilon',\delta') + \frac{|Q|}{2\varepsilon'^2}\log\frac{2}{\delta'}. \end{aligned}$$

From this one can obtain that $C(S,\varepsilon',\delta') = O\Big(\frac{|S|\log|S|}{\varepsilon'^2}\log\frac{1}{\delta'}\Big)$. Hence,

$$C\left(|S|,\frac{\varepsilon}{\log|S|},\frac{\delta}{|S|^2}\right) = O\left(\frac{|S|\log^3|S|}{\varepsilon^2}\log\frac{|S|^2}{\delta}\right).$$

Now we bound the error. By Lemma 17, with probability $\geq 1-|Q|\delta$,

$$\begin{aligned} & err(\text{MERGE-RANK}(Q,\varepsilon',\delta')) \leq \\ & \max\{err\big(\text{MERGE-RANK}\big(Q[1:\lfloor |Q|/2 \rfloor],\varepsilon',\delta'\big)\big), \\ & err\big(\text{MERGE-RANK}\big(T[\lfloor |Q|/2 \rfloor + 1 : |Q|],\varepsilon',\delta'\big)\big)\} + \varepsilon'. \end{aligned} \tag{2.5}$$

We can bound the total times MERGE is called in a single instance of MERGE-RANK(S,ε',δ'). MERGE combines the singleton sets and forms the sets with two elements, it combines the sets with two elements and forms the sets with four elements and henceforth. Hence the total times MERGE is called is $\sum_{i=1}^{\log|S|}\frac{|S|}{2^i} \leq |S|$. Therefore, the probability that Equation 2.5 holds every time when two ordered sets are merged in MERGE-RANK(S,ε',δ') is $\leq |S|\cdot|S|\delta' = |S|^2\delta'$.

If Equation 2.5 holds every time MERGE is called, then error of MERGE-RANK(S,ε',δ') is at most $\sum_{i=1}^{\log|S|}\varepsilon' \leq \varepsilon'\log|S|$. This is because $err(S)$ is 0 if S has only one element. And a singleton set participates in $\log n$ merges before becoming the final output set.

Therefore, w.p. $\geq 1-|S|^2\delta'$,

$$err(\textsc{Merge-Rank}(S,\varepsilon',\delta')) \leq \log|S|\varepsilon'.$$

Hence with probability $\geq 1-\delta$,

$$err\left(\textsc{Merge-Rank}\left(S,\frac{\varepsilon}{\log|S|},\frac{\delta}{|S|^2}\right)\right) \leq \varepsilon. \qquad \square$$

2.F Proofs for Section 2.4.2

Proof of Lemma 6

Proof. Let set S be ordered s.t. $\tilde{p}(S(i),S(j)) \geq 0\ \forall i > j$. Let $S''_k = \{S(l) : k \leq l \leq k+5(\log n)^{x+1}-1\}$. The probability that none of the elements in S''_k is selected for a given k is

$$\leq \left(1-\frac{5(\log n)^{x+1}}{n}\right)^{n/(\log n)^x} < \frac{1}{n^5}.$$

Therefore by union bound, the probability that none of the elements in S''_k is selected for any k is

$$\leq n\cdot\frac{1}{n^5} = \frac{1}{n^4}. \qquad \square$$

Proof of Lemma 8

We prove Lemma 8 by dividing it into smaller lemmas. We refer to $|\tilde{p}(e,f)|$ as a measure of distance between elements e and f.

We divide all elements in S into two sets based on distance from anchors. First set contains all elements that are far away from all anchors and the second set contains all elements which are close to atleast one of the anchors. INTERVAL-BINARY-SEARCH acts differently on both sets.

We first show that for elements in the first set, INTERVAL-BINARY-SEARCH places them in between the right anchors by using just the random walk subroutine.

For elements in the second set, INTERVAL-BINARY-SEARCH might fail to find the right anchors just by using the random walk subroutine. But we show that INTERVAL-BINARY-SEARCH visits a close anchor during random walk and BINARY-SEARCH finds a close anchor from the set of visited anchors using simple binary search.

We first prove Lemma 8 for the elements of first set.

Lemma 20. *For $\varepsilon'' > \varepsilon'$, consider an ε'-ranked S'. If an element e is such that $|\tilde{p}(e,S'(j))| > \varepsilon''$ $\forall j$, then with probability $\geq 1 - \frac{1}{n^6}$ step 4a of* INTERVAL-BINARY-SEARCH(S',e,ε'') *outputs the index y such that $\tilde{p}(e,S'(y)) > \varepsilon''$ and $\tilde{p}(S'(y+1),e) > \varepsilon''$.*

Proof. We first show that there is an unique y s.t. $\tilde{p}(e,S'(y)) > \varepsilon''$ and $\tilde{p}(S'(y+1),e) > \varepsilon''$.

Let i be the largest index such that $\tilde{p}(e,S'(i)) > \varepsilon''$. By Lemma 19, $\tilde{p}(e,S'(j)) > \varepsilon'' - \varepsilon' > 0 \quad \forall j < i$. Hence by the assumption on e, $\tilde{p}(e,S'(j)) > \varepsilon'' \quad \forall j < i$. Let k be the smallest index such that $\tilde{p}(S'(k),e) > \varepsilon''$. By a similar argument as previously, we can show that $\tilde{p}(S'(j),e) > \varepsilon'' \quad \forall j > k$.

Hence by the above arguments and the fact that $|\tilde{p}(e,S'(j))| > \varepsilon'' \quad \forall j$, there exists only one y such that $\tilde{p}(e,S'(y)) > \varepsilon''$ and $\tilde{p}(S'(y+1),e) > \varepsilon''$.

Thus in the tree T, there is only one leaf node w such that $\tilde{p}(e,S'(w_1)) > \varepsilon''$ and $\tilde{p}(S'(w_2),e) > \varepsilon''$.

Consider some node m which is not an ancestor of w. Then either $\tilde{p}(S'(m_1),e) > \varepsilon''$ or $\tilde{p}(S'(m_2),e) < -\varepsilon''$. Since we compare e with $S'(m_1)$ and $S'(m_2)$ $\frac{10}{\varepsilon''^2}$ times, we move to the parent of m with probability atleast $\frac{19}{20}$.

Consider some node m which is an ancestor of w. Then $\tilde{p}(S'(m_1),e) < -\varepsilon''$, $\tilde{p}(S'(m_2),e) > \varepsilon''$, and $|\tilde{p}(S'(\lceil \frac{m_1+m_2}{2} \rceil),e)| > \varepsilon''$. Therefore we move in direction of w with probability atleast $\frac{19}{20}$.

Therefore if we are not at w, then we move towards w with probability atleast $\frac{19}{20}$ and if we are at w then the count c increases with probability atleast $\frac{19}{20}$.

Since we start at most $\log n$ away from w if we move towards w for $21\log n$ then the algorithm will output y. The probability that we will move towards w less than $21\log n$ times is $\leq e^{-30\log nD(\frac{21}{30}||\frac{19}{20})} \leq \frac{1}{n^6}$. □

To prove Lemma 8 for the elements of the second set, we first show that the random walk subroutine of algorithm INTERVAL-BINARY-SEARCH placing an element in wrong bin is highly unlikely.

Lemma 21. *For $\varepsilon'' > \varepsilon'$, consider an ε'-ranked set S'. Now consider an element e and y such that either $\tilde{p}(S'(y),e) > \varepsilon''$ or $\tilde{p}(S'(y+1),e) < -\varepsilon''$, then step 4a of* INTERVAL-BINARY-SEARCH(S',e,ε'') *will not output y with probability $\geq 1-\frac{1}{n^7}$.*

Proof. Recall that step 4a of INTERVAL-BINARY-SEARCH outputs y if we are at the leaf node $(y,y+1)$ and the count c is atleast $10\log n$.

Since either $\tilde{p}(S'(y),e) > \varepsilon''$ or $\tilde{p}(S'(y+1),e) < -\varepsilon''$, when we are at leaf node $(y,y+1)$, the count decreases with probability atleast $\frac{19}{20}$. Hence the probability that INTERVAL-BINARY-SEARCH is at $(y,y+1)$ and the count is greater than $10\log n$ is at most $\sum_{i=10\log n}^{30\log n} e^{-i\cdot D(\frac{i-10\log n}{2i}||\frac{19}{20})} < 20\log n e^{-10\log nD(\frac{1}{3}||\frac{19}{20})} \leq \frac{1}{n^7}$. □

We now show that for an element of the second set, the random walk subroutine either places it in correct bin or visits a close anchor.

Lemma 22. *For $\varepsilon'' > \varepsilon'$, consider an ε'-ranked set S'. Now consider an element e that is close to an element in S' i.e., $\exists g : |\tilde{p}(S'(g),e)| < \varepsilon''$. With probability $\geq 1-\frac{1}{n^6}$, step 4a of* INTERVAL-BINARY-SEARCH(S',e,ε'') *will either output the right index y such that $\tilde{p}(S'(y),e) < \varepsilon''$ and $\tilde{p}(S'(y+1),e) > -\varepsilon''$ or* INTERVAL-BINARY-SEARCH *visits $S'(h)$ such that $|\tilde{p}(S'(h),e)| < 2\varepsilon''$.*

Proof. By Lemma 21, step 4a of INTERVAL-BINARY-SEARCH does not output a wrong interval with probability $1-\frac{1}{n^7}$. Hence we just need to show that w.h.p., e visits a close anchor.

Let i be the largest index such that $\tilde{p}(e,S'(i)) > 2\varepsilon''$. Then $\forall j < i$, by Lemma 19, $\tilde{p}(e,S(j)) > 2\varepsilon'' - \varepsilon' > \varepsilon''$.

Let k be the smallest index such that $\tilde{p}(S'(k),e) > 2\varepsilon''$. Then $\forall j > k$, by Lemma 19, $\tilde{p}(S'(j),e) > \varepsilon''$.

Therefore for $u < v$ such that $\min(|\tilde{p}(S'(u),e)|, |\tilde{p}(S'(v),e)|) \geq 2\varepsilon''$ only one of three sets $\{x : x < u\}$,$\{x : u < x < v\}$ and $\{x : x > v\}$ contains an index z such that $|\tilde{p}(S'(z),e)| < \varepsilon''$.

Let a node α be s.t. for some $c \in \{\alpha_1, \alpha_2, \lceil \frac{\alpha_1+\alpha_2}{2} \rceil\}$, $|\tilde{p}(S'(c),e)| \leq 2\varepsilon''$. If INTERVAL-BINARY-SEARCH reaches such a node α then we are done.

So assume that INTERVAL-BINARY-SEARCH is at a node β s.t. $\forall c \in \{\beta_1, \beta_2, \lceil \frac{\beta_1+\beta_2}{2} \rceil\}$, $|\tilde{p}(S'(c),e)| > 2\varepsilon''$. Note that only one of three sets $\{x : x < \beta_1 \text{ or } x > \beta_2\}$, $\{x : \beta_1 < x < \lceil \frac{\beta_1+\beta_2}{2} \rceil\}$ and $\{x : \lceil \frac{\beta_1+\beta_2}{2} \rceil < x < \beta_2\}$ contains an index z such that $|\tilde{p}(S'(z),e)| < \varepsilon''$ and INTERVAL-BINARY-SEARCH moves towards that set with probability $\frac{19}{20}$. Hence the probability that we never visit an anchor that is less than $2\varepsilon''$ away is at most $e^{-30\log n D(\frac{15.5}{30} || \frac{19}{20})} \leq \frac{1}{n^7}$. □

We now complete the proof by showing that for an element e from the second set, if Q contains an index y of an anchor that is close to e, BINARY-SEARCH will output one such index.

Lemma 23. *For $\varepsilon'' > \varepsilon'$, consider ordered sets S', Q s.t. $p(S'(Q(i)), S'(Q(j))) > \frac{1}{2} - \varepsilon'$ $\forall i > j$. For an element e s.t., $\exists g : |\tilde{p}(S'(Q(g)),e)| < 2\varepsilon''$, BINARY-SEARCH$(S', Q, e, \varepsilon'')$ will return y such that $|\tilde{p}(S'(Q(y)),e)| < 4\varepsilon''$ with probability $\geq 1 - \frac{1}{n^6}$.*

Proof. At any stage of BINARY-SEARCH, there are three possibilities that can happen . Consider the case when we are comparing e with $S'(Q(i))$.

1. $|\tilde{p}(S'(Q(i)),e)| < 2\varepsilon''$. Probability that the fraction of wins for e is not between $\frac{1}{2} - 3\varepsilon''$ and $\frac{1}{2} + 3\varepsilon''$ is less than $e^{-\frac{10\log n}{\varepsilon''^2}\varepsilon''^2} \leq \frac{1}{n^{10}}$. Hence BINARY-SEARCH outputs $Q(i)$.

2. $\tilde{p}(S'(Q(i)),e) > 2\varepsilon''$. Probability that the fraction of wins for e is more than $\frac{1}{2}$ is less than $e^{-\frac{10\log n}{\varepsilon''^2}\varepsilon''^2} \leq \frac{1}{n^{10}}$. So BINARY-SEARCH will not move right. Also notice that $\tilde{p}(S'(Q(j)),e) > 2\varepsilon'' - \varepsilon' > \varepsilon''$ $\forall j > i$.

3. $\tilde{p}(S'(Q(i)),e) > 4\varepsilon''$. Probability that the fraction of wins for e is more than $\frac{1}{2} - 3\varepsilon''$ is less than $e^{-\frac{10\log n}{\varepsilon''^2}\varepsilon''^2} \leq \frac{1}{n^{10}}$. Hence BINARY-SEARCH will move left. Also notice that $\tilde{p}(S'(Q(j)),e) > 4\varepsilon'' - \varepsilon' > \varepsilon'' \; \forall j > i$.

We can show similar results for $\tilde{p}(S'(Q(i)),e) < -2\varepsilon''$ and $\tilde{p}(S'(Q(i)),e) < -4\varepsilon''$. Hence if $|\tilde{p}(S'(Q(i)),e)| < 2\varepsilon''$ then BINARY-SEARCH outputs $Q(i)$, and if $2\varepsilon'' < |\tilde{p}(S'(Q(i)),e)| < 4\varepsilon''$ then either BINARY-SEARCH outputs $Q(i)$ or moves in the correct direction and if $|\tilde{p}(S'(Q(i)),e)| > 4\varepsilon''$, then BINARY-SEARCH moves in the correct direction. □

Lemma 24. INTERVAL-BINARY-SEARCH(S,e,ε) *terminates in* $O(\frac{\log n \log\log n}{\varepsilon^2})$ *comparisons for any set* S *of size* $O(n)$.

Proof. Step 3 of INTERVAL-BINARY-SEARCH runs for $30\log n$ iterations. In each iteration, INTERVAL-BINARY-SEARCH compares e with at most 3 anchors and repeats each comparison for $10/\varepsilon^2$. So total comparisons in step 3 is $O(\log n/\varepsilon^2)$. The size of Q is upper bounded by $90\log n$ and BINARY-SEARCH does a simple binary search over Q by repeating each comparison $10\log n/\varepsilon^2$. Hence total comparisons used by BINARY-SEARCH is $O(\log n \log\log n/\varepsilon^2)$ □

Combining Lemmas 7, 21, 22, 23, 24 yields the result.

Proof of Lemma 11

Proof. Combining Lemmas 7, 10 and using union bound, at the end of step 5a ,w.p. $\geq 1 - \frac{2}{n^3}$, S' is ε'-ranked and $\forall j, e \in B_j$, $\min(\tilde{p}(e,S'(j)), \tilde{p}(S'(j+1),e)) > 5\varepsilon''$. Hence by Lemma 19, $\forall j,k < j, e \in B_j$, $\tilde{p}(e,S'(k)) > 5\varepsilon'' - \varepsilon' > 4\varepsilon''$. Similarly, $\forall j,k > j, e \in B_j$, $\tilde{p}(S'(k),e) > 5\varepsilon'' - \varepsilon' > 4\varepsilon''$.

If $|B_j| > 0$, then $\tilde{p}(e,S'(k)) > 4\varepsilon''$ for $e \in B_j, k \leq j$, $\tilde{p}(S'(l),e) > 4\varepsilon''$ for $e \in B_j, l \geq j+1$. Hence by stochastic transitivity, $\tilde{p}(S'(l),S'(k)) > 4\varepsilon''$ for $l > j \geq k$. Therefore there exists k,l s.t. $\tilde{p}(S'(l),f) > 0 \; \forall f \in \{S'(y) : y \leq j\}$, $\tilde{p}(S'(k),S'(l)) > 0$ and $\tilde{p}(f,S'(k)) > 0 \; \forall f \in \{S'(y) : y > j\}$. Now by Lemma 6, w.p. $\geq 1 - \frac{1}{n^4}$, size of all such sets B_j is less than $5(\log n)^{x+1}$.

Lemma follows by union bound. □

Proof of Theorem 13

We first bound the running time of BINARY-SEARCH-RANKING algorithm.

Theorem 25. BINARY-SEARCH-RANKING *terminates after* $O(\frac{n(\log\log n)^x}{\varepsilon^2}\log n)$ *comparisons with probability* $\geq 1-\frac{1}{n^2}$.

Proof. Step 2 RANK-$x(S',\varepsilon',\frac{1}{n^6})$ terminates after $O(\frac{n}{\varepsilon^2}\log n)$ comparisons with probability $\geq 1-\frac{1}{n^6}$.

By Lemma 8, for each element e, the step 4a INTERVAL-BINARY-SEARCH(S',e,ε'') terminates after $O(\frac{\log n\log\log n}{\varepsilon^2})$ comparisons. Hence step 4 takes at most $O(\frac{n\log n\log\log n}{\varepsilon^2})$ comparisons.

Comparing each element with the anchors in steps 5a takes at most $O(\frac{\log n}{\varepsilon^2})$ comparisons.

With probability $\geq 1-\frac{1}{n^4}$ step 5b RANK-X$(B_i,\ \varepsilon'',\frac{1}{n^4})$ terminates after $O(|B_i|\frac{(\log|B_i|)^x}{\varepsilon^2}\log n)$ comparisons. By Lemma 11, $|B_i|\leq 5(\log n)^{x+1}$ for all i w.p. $\geq 1-\frac{3}{n^3}$. Hence, w.p. $\geq 1-\frac{3}{n^3}$, total comparisons to rank all B_is is at most $\sum_i O(|B_i|\frac{(\log|B_i|)^x}{\varepsilon^2}\log n)\leq\sum_i O(\frac{|B_i|\log n(\log(5(\log n)^{x+1}))^x}{\varepsilon^2})=O(\frac{n\log n(\log\log n)^x}{\varepsilon^2})$.

Therefore, by summing comparisons over all steps, with probability $\geq 1-\frac{1}{n^2}$ total comparisons is at most $O\left(\frac{n\log n(\log\log n)^x}{\varepsilon^2}\right)$. □

Now we show that BINARY-SEARCH-RANKING outputs an ε-ranking with high probability.

Theorem 26. BINARY-SEARCH-RANKING *produces an* ε*-ranking with probability at least* $1-\frac{1}{n^2}$.

Proof. By combining Lemmas 7, 9, 10, 12 and using union bound, w.p. $\geq 1-\frac{1}{n^2}$, at the end of step 5b,

- S' is ε'-ranked.

- Each C_i has elements such that $|\tilde{p}(C_i(j), S(i))| < 7\varepsilon''$ for all j.
- Each B_i has elements such that $\tilde{p}(S'(i), B_i(j)) < -5\varepsilon''$ and $\tilde{p}(S'(i+1), B_i(j)) > 5\varepsilon''$ for all j.
- All B_is are ε''-ranked.

For $j \geq i$, $e \in B_{i-1} \bigcup S'(i) \bigcup C_i$, $f \in S'(j) \bigcup C_j \bigcup B_j$,
$\tilde{p}(e,f) \leq \tilde{p}(e,S'(i)) + \tilde{p}(S'(i),S'(j)) + \tilde{p}(S'(j),f) \leq 7\varepsilon'' + \varepsilon' + 7\varepsilon'' < 15\varepsilon'' = \varepsilon$. Combining the above results proves the Theorem. □

Combining Theorems 25, 26 yields the result.

Proof Sketch for Theorem 16

Proof sketch. Consider a stochastic model where there is an inherent ranking r and for any two consecutive elements $p(i,i+1) = \frac{1}{2} - 2\varepsilon$. Suppose there is a genie that knows the true ranking r up to the sets $\{r(2i-1), r(2i)\}$ for all i i.e., for each i, genie knows $\{r(2i-1), r(2i)\}$ but it does not know the ranking between these two elements. Since consecutive elements have $\varepsilon(i,i+1) = 2\varepsilon > \varepsilon$, to find an ε-ranking, the genie has to correctly identify the ranking within all the $n/2$ pairs. Using Fano's inequality from information theory, it can be shown that the genie needs at least $\Omega\left(\frac{n}{\varepsilon^2} \log \frac{n}{\delta}\right)$ comparisons to identify the ranking of the consecutive elements with probability $1-\delta$. □

2.G Additional Experiments

As we mentioned in Section 5.5, **BTM-PAC** allows comparison of an element with itself. It is not beneficial when the goal is to find ε-maximum. So we modify their algorithm by not allowing such comparisons. We refer to this restricted version as **R-BTM-PAC**.

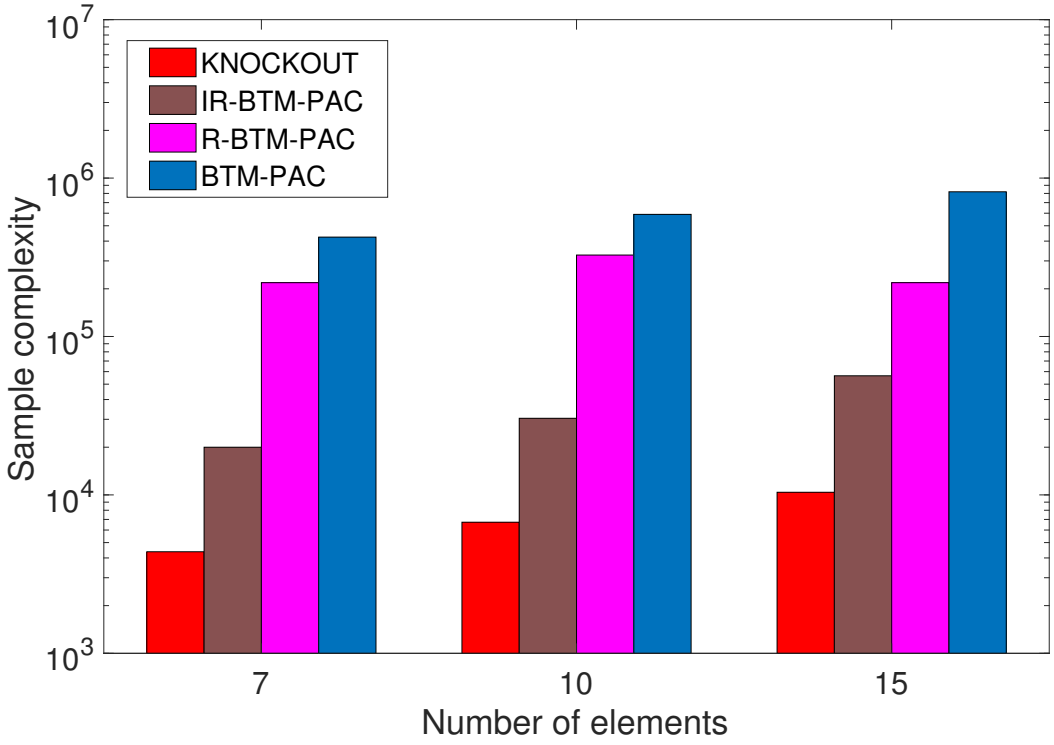


Figure 2.6: Sample complexity comparison of KNOCKOUT and variations of BTM-PAC for different input sizes, with $\varepsilon = 0.05$ and $\delta = 0.1$

As seen in figure, performance of **BTM-PAC** does not increase by much by restricting the comparisons.

We further reduce the constants in **R-BTM-PAC**. We change Equations (7) and (8) in [YJ11] to $c_\delta(t) = \sqrt{\frac{1}{t}\log\frac{n^3 N}{\delta}}$ and $N = \lceil \frac{1}{\varepsilon^2}\log\frac{n^3 N}{\delta} \rceil$, respectively.

We believe the same guarantees hold even with the updated constants. We refer to this improved restricted version as **IR-BTM-PAC**. Here too we consider the stochastic model where $p(i,j) = 0.6 \forall\ i < j$ and we find 0.05-maximum with error probability $\delta = 0.1$.

In Figure 2.6 we compare the performance of KNOCKOUT and all variations of BTM-PAC. As the figure suggests, the performance of **IR-BTM-PAC** improves a lot but KNOCKOUT still outperforms it significantly.

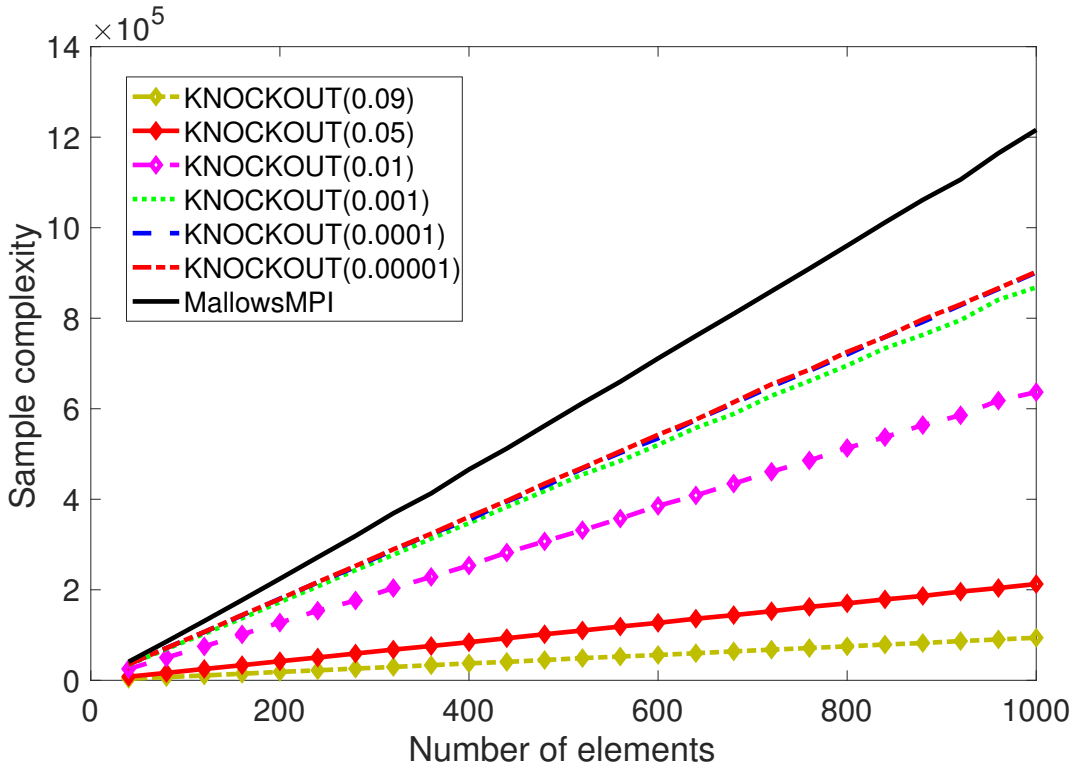


Figure 2.7: Sample complexity of KNOCKOUT for different values of n and ε

In Figure 2.7, we consider the stochastic model where $p(i,j) = 0.6\ \forall i < j$ and find ε-maximum for different values of ε. Similar to previous experiments, we use $\delta = 0.1$. As we can see the number of comparisons increases almost linearly with n. Further the number of comparisons does not increase significantly even when ε decreases. Also the number of comparisons seem to be converging as ε goes to 0. KNOCKOUT outperforms **MallowsMPI** even for the very small ε values. We attribute this to the subroutine COMPARE that finds the winner faster when the distance between elements are much larger than ε.

Chapter 3

Maxing and Ranking with Few Assumptions

3.1 Introduction

3.1.1 Motivation

Maximum selection (maxing) and sorting using pairwise comparisons are among the most practical and fundamental algorithmic problems in computer science. As is well-known, maxing requires $n-1$ comparisons, while sorting takes $\Theta(n \log n)$ comparisons.

The probabilistic version of this problem, where comparison outcomes are random, is of significant theoretical interest as well, and it too arises in many applications and diverse disciplines. In sports, pairwise games with random outcomes are used to determine the best, or the order, of teams or players. Similarly *Trueskill* [HMG06] matches video gamers to create their ranking. It is also used for a variety of online applications such as to learn consumer preferences with the popular *A/B tests*, in recommender systems [WSE14], for ranking documents from user clickthrough data [RJ07, RKJ08], and more. The popular crowd sourcing website GIFGIF [gif] shows how pairwise comparisons can help associate emotions with many animated GIF images.

Visitors are presented with two images and asked to select the one that better corresponds to a given emotion. For these reasons, and because of its intrinsic theoretical interest, the problem received a fair amount of attention.

3.1.2 Terminology and previous results

One of the first studies in the area, [FRPU94] assumed n totally-ordered elements, where each comparison errs with the same, known, probability $\alpha < \frac{1}{2}$. It presented a maxing algorithm that uses $O(\frac{n}{\alpha^2}\log\frac{1}{\delta})$ comparisons to output the maximum with probability $\geq 1-\delta$, and a ranking algorithm that uses $O(\frac{n}{\alpha^2}\log\frac{n}{\delta})$ comparisons to output the ranking with probability $\geq 1-\delta$.

These results have been and continue to be of great interest. Yet this model has two shortcomings. It assumes that there is only one random comparison probability, α, and that its value is known. In practice, comparisons have different, and arbitrary, probabilities, and they are not known in advance. To address more realistic scenarios, researchers considered more general probabilistic models.

Consider a set of n elements, without loss of generality $[n] \stackrel{\text{def}}{=} \{1,2,\ldots,n\}$. A *probabilistic model*, or *model* for short, is an assignment of a *preference probability* $p_{i,j} \in [0,1]$ for every $i \neq j \in [n]$, reflecting the probability that i is *preferred* when compared with j. We assume that repeated comparisons are independent and that there are no "draws", hence $p_{j,i} = 1 - p_{i,j}$.

If $p_{i,j} \geq \frac{1}{2}$, we say that i is *preferable* to j and write $i \geq j$. Element i is *maximal* in a model if $i \geq j$ for all $j \neq i$. And a permutation $\ell_1,\ldots,\ell_n$ is a *ranking* if $\ell_i \geq \ell_j$ for all $i \leq j$. Observe that the first element of any ranking is always maximal. For example, for $n = 3$, $p_{1,2} = 1/2$, $p_{1,3} = 1/3$, and $p_{2,3} = 2/3$, we have $1 \geq 2$, $2 \geq 1$, $3 \geq 1$, and $2 \geq 3$. Hence 2 is the unique maximum, and 2,3,1 is the unique ranking. We seek algorithms that without knowing the underlying model, use pairwise comparisons to find a maximal element and a ranking.

Two concerns spring to mind. First, there may be two elements i, j with $p_{i,j}$ arbitrarily close to half, requiring arbitrarily many comparisons just to determine which is preferable

to the other. This concern has a common remedy, that we also adopt. The *PAC* paradigm, e.g. [YJ11, BFSH14], that requires the algorithm's output to be only *Probably Approximately Correct*.

Let $\tilde{p}_{i,j} \stackrel{\text{def}}{=} p_{i,j} - \frac{1}{2}$ be the *centered* preference probability. Note that $\tilde{p}_{i,j} \geq 0$ iff i is preferable to j. If $\tilde{p}_{i,j} \geq -\varepsilon$ we say that i is ε-preferable to j. For $0 < \varepsilon < 1/2$, an element $i \in [n]$ is ε-*maximum* if it is ε-preferable to all other elements, namely, $\tilde{p}_{i,j} \geq -\varepsilon \; \forall j \neq i$. Given $\varepsilon > 0, \frac{1}{2} \geq \delta > 0$, a PAC maxing algorithm must output an ε-maxima with probability $\geq 1-\delta$, henceforth abbreviated *with high probability (WHP)*. Similarly, a permutation $\ell_1, \ldots, \ell_n$ of $\{1, \ldots, n\}$ is an ε-*ranking* if ℓ_i is ε-preferable to ℓ_j for all $i \leq j$, and a PAC ranking algorithm must output an ε-ranking WHP. Note that in this paper, we consider $\delta \leq \frac{1}{2}$, the more practical regime. For larger values of δ, one can use our algorithms with $\delta = \frac{1}{2}$.

The second concern is that not all models have a ranking, or even a maximal element. For example, for $p_{1,2} = p_{2,3} = p_{3,1} = 1$, or the more opaque yet interesting non-transitive coins [Wik17], each element is preferable to the cyclically next, hence there is no maximal element and no ranking.

A standard approach, that again we too will adopt, to address this concern is to consider structured models. The simplest may be parametric models, of which one of the more common is *Placket Luce* (PL) [Pla75, Luc05], where each element i is associated with an unknown positive number a_i and $p_{i,j} = \frac{a_i}{a_i + a_j}$. [SBFPH15] derived a PAC maxing algorithm that uses $O(\frac{n}{\varepsilon^2} \log \frac{n}{\varepsilon\delta})$ comparisons and a PAC ranking algorithm that uses $O(\frac{n}{\varepsilon^2} \log n \log \frac{n}{\varepsilon\delta})$ comparisons for any PL model. Related results for the *Mallows model* under a non-PAC paradigm were derived by [BFHS14].

But significantly more general, and more realistic, non-parametric, models may also have maxima and rankings. A model is *strongly stochastically transitive (SST)*, if $i \geq j$ and $j \geq k$ imply $p_{i,k} \geq \max(p_{i,j}, p_{j,k})$. By simple induction, every SST model has a maximum element and a ranking. And one additional property, that is perhaps more difficult to justify, has proved

helpful in constructing maxing and sorting PAC algorithms. A tournament satisfies the *stochastic triangle inequality* if $i \geq j$ and $j \geq k$ imply that $\tilde{p}_{i,k} \leq \tilde{p}_{i,j} + \tilde{p}_{j,k}$.

In Section 4.5 we show that if a model has a ranking, then an ε-ranking can be found WHP via $O(\frac{n^2}{\varepsilon^2}\log\frac{n}{\delta})$ comparisons. For all models that satisfy both SST and triangle inequality, [YJ11] derived a PAC maxing algorithm that uses $O(\frac{n}{\varepsilon^2}\log\frac{n}{\varepsilon\delta})$ comparisons. [FOPS17] eliminated the $\log\frac{n}{\varepsilon}$ factor and showed that $O\left(\frac{n}{\varepsilon^2}\log\frac{1}{\delta}\right)$ comparisons suffice and are optimal, and constructed a nearly-optimal PAC ranking algorithm that uses $O(\frac{n\log n(\log\log n)^3}{\varepsilon^2})$ comparisons for all $\delta \geq \frac{1}{n}$, off by a factor of $O((\log\log n)^3)$ from optimum. Lower-bounds follow from an analogy to [ZCL14, FRPU94]. Observe that since the PL model satisfies both SST and triangle inequality, these results also improve the corresponding PL results.

Finally, we consider models that are not SST, or perhaps don't have maximal elements, rankings, or even their ε-equivalents. In all these cases, one can apply a weaker order relation. The *Borda score* $s(i) \stackrel{\text{def}}{=} \frac{1}{n}\sum_j p_{i,j}$ is the probability that i is preferable to another, randomly selected, element. Element i is *Borda maximal* if $s(i) = \max_j s(j)$, and ε-*Borda maximal* if $s(i) \geq \max_j s(j) - \varepsilon$. A PAC Borda-maxing algorithm outputs an ε-Borda maximal element WHP (with probability $\geq 1-\delta$). Similarly, a *Borda ranking* is a permutation $i_1,\ldots,i_n$ such that for all $1 \leq j \leq n-1$, $s(i_j) \geq s(i_{j+1})$. An ε-*Borda ranking* is a permutation where for all $1 \leq j \leq k \leq n$, $s(i_j) \geq s(i_k) - \varepsilon$. A PAC Borda-ranking algorithm outputs an ε-Borda ranking WHP.

Recall that Borda scores apply to all models. As noted in [HSRW16, UCFN13, BFSH14, JKDN15] considering elements with nearly identical Borda scores shows that exact Borda-maxing and ranking requires arbitrarily many comparisons. [BFSH14] derived a PAC Borda ranking, and therefore also maxing, algorithms that use $O(\frac{n^2}{\varepsilon^2})$ comparisons. [LGAL14] derived a $O(\frac{n\log n}{\varepsilon^2}\log(\frac{n}{\delta}))$ PAC Borda ranking algorithm for restricted setting. However note that several simple models, including $p_{1,2} = p_{2,3} = p_{3,1} = 1$ do not belong to this model.

[AJOS14b, AFJ$^+$16, AFHN15] considered deterministic adversarial versions of this problem that has applications in [AJOS14a]. Finally, we note that all our algorithms are adaptive,

where each comparison is chosen based on the outcome of previous comparisons. Non-adaptive algorithms were discussed in [RA14, NOS12, NOS16, JKSO16].

3.2 Results and Outline

Our goal is to find the minimal assumptions that enable efficient algorithms for these problems. In particular, we would like to see if we can eliminate the somewhat less-natural triangle inequality. With two algorithmic problems: maxing and ranking, and one property–SST and one special metric–Borda scores, the puzzle consists of four main questions.

1) With just SST (and no triangle inequality) are there: a) PAC maxing algorithms with $O(n)$ comparisons? b) PAC ranking algorithms with near $O(n\log n)$ comparisons? 2) With no assumptions at all, but for the Borda-score metric, are there: a) PAC Borda-maxing algorithms with $O(n)$ comparisons? b) PAC Borda-ranking algorithms with near $O(n\log n)$ comparisons?

We essentially resolve all four questions. 1a) Yes. In Section 3.3, Theorem 32, we use SST alone to derive a $O\left(\frac{n}{\varepsilon^2}\log\frac{1}{\delta}\right)$ comparisons PAC maxing algorithm. Note that this is the same complexity as with triangle inequality, and it matches the lower bound. 1b) No. In Section 4.5, Theorem 33, we show that there are SST models where any PAC ranking algorithm with $\varepsilon \leq 1/4$ requires $\Omega(n^2)$ comparisons. This is significantly higher than the roughly $O(n\log n)$ comparisons needed with triangle inequality, and is close to the $O(n^2\log n)$ comparisons required without any assumptions. 2a) Yes. In Section 3.5, Theorem 34, we derive a PAC Borda maxing algorithm that without any model assumptions requires $O\left(\frac{n}{\varepsilon^2}\log\frac{1}{\delta}\right)$ comparisons which is order optimal. 2b) Yes. In Section 3.5, Theorem 35, we derive a PAC Borda ranking algorithm that without any model assumptions requires $O\left(\frac{n}{\varepsilon^2}\log\frac{n}{\delta}\right)$ comparisons.

Beyond the theoretical results sections, in Section 4.8, we provide experiments on simulated data. In Section 5.6, we discuss the results.

3.3 Maxing

3.3.1 SEQ-ELIMINATE

Our main building block is a simple, though sub-optimal, algorithm SEQ-ELIMINATE that sequentially eliminates one element from input set to find an ε-maximum under SST.

SEQ-ELIMINATE uses $O\left(\frac{n}{\varepsilon^2}\log\frac{n}{\delta}\right)$ comparisons and w.p.$\geq 1-\delta$, finds an ε-maximum. Even for simpler models [ZCL14] we know that an algorithm needs $\Omega\left(\frac{n}{\varepsilon^2}\log\frac{1}{\delta}\right)$ comparisons to find an ε-maximum w.p.$\geq 1-\delta$. Hence the number of comparisons used by SEQ-ELIMINATE is optimal up to a constant factor when $\delta \leq \frac{1}{n}$ but can be $\log n$ times the lower bound for $\delta = \frac{1}{2}$.

By SST, any element that is ε-preferable to absolute maximum element of S is an ε-maximum of S. Therefore if we can reduce S to a subset S' of size $O(\frac{n}{\log n})$ that contains an absolute maximum of S using $O\left(\frac{n}{\varepsilon^2}\log\frac{1}{\delta}\right)$ comparisons, we can then use SEQ-ELIMINATE to find an ε-maximum of S' and the number of comparisons is optimal up to constants. We provide one such reduction in subsection 3.3.2.

Sequential elimination techniques have been used before [BFHS14] to find an absolute maximum. In such approaches, a running element is maintained, and is compared and replaced with a competing element in S if the latter is found to be better with confidence $\geq 1-\delta/n$. Note that if the running and competing elements are close to each other, this technique can take an arbitrarily long time to declare the winner. But since we are interested in finding only an ε-maximum, SEQ-ELIMINATE circumvents this issue. We later show that SEQ-ELIMINATE needs to update the running element r with the competing element c if $\tilde{p}_{c,r} \geq \varepsilon$ and retain r if $\tilde{p}_{c,r} \leq 0$. If $0 < \tilde{p}_{c,r} < \varepsilon$, replacing or retaining r doesn't affect the performance of SEQ-ELIMINATE significantly. Thus, in other words we've reduced the problem to testing whether $\tilde{p}_{c,r} \leq 0$ or $\tilde{p}_{c,r} \geq \varepsilon$.

Assuming that testing problem always returns the right answer, since SEQ-ELIMINATE never replaces the running element with a worse element, either the output is the absolute

maximum b^* or b^* is never the running element. If b^* is eliminated against running element r then $\tilde{p}_{b^*,r} \leq \varepsilon$ and hence r is an ε-maximum and since the running element only gets better, the output is an ε-maximum.

We first present a testing procedure COMPARE that we use to update the running element in SEQ-ELIMINATE.

COMPARE

COMPARE$(i, j, \varepsilon_l, \varepsilon_u, \delta)$ takes two elements i and j, and two biases $\varepsilon_u > \varepsilon_l$, and with confidence $\geq 1-\delta$, determines whether $\tilde{p}_{i,j}$ is $\leq \varepsilon_l$ or $\geq \varepsilon_u$.

For this, COMPARE compares the two elements $2/(\varepsilon_u - \varepsilon_l)^2 \log(2/\delta)$ times. Let $\hat{p}_{i,j}$ be the fraction of times i beats j, and let $\hat{\tilde{p}}_{i,j} \stackrel{\text{def}}{=} \hat{p}_{i,j} - \frac{1}{2}$. If $\hat{\tilde{p}}_{i,j} < (\varepsilon_l + \varepsilon_u)/2$, COMPARE declares $\tilde{p}_{i,j} \leq \varepsilon_l$ (returns 1), and otherwise it declares $\tilde{p}_{i,j} \geq \varepsilon_u$ (returns 2).

Due to lack of space, we present the algorithm COMPARE in Appendix 5.B along with certain improvements for better performance in practice .

In the Lemma below, we bound the number of comparisons used by COMPARE and prove its correctness. Proof is in 3.A.2.

Lemma 27. *For $\varepsilon_u > \varepsilon_l$,* COMPARE$(i, j, \varepsilon_l, \varepsilon_u, \delta)$ *uses $\leq \frac{2}{(\varepsilon_u - \varepsilon_l)^2} \log \frac{2}{\delta}$ comparisons and if $\tilde{p}_{i,j} \leq \varepsilon_l$, then w.p. $\geq 1-\delta$, it returns* 1*, else if $\tilde{p}_{i,j} \geq \varepsilon_u$, w.p. $\geq 1-\delta$, it returns* 2*.*

Now we present SEQ-ELIMINATE that uses the testing subroutine COMPARE and finds an ε-maximum.

SEQ-ELIMINATE Algorithm

SEQ-ELIMINATE takes a variable set S, selects a random *running element* $r \in S$ and repeatedly uses COMPARE$(c, r, 0, \varepsilon, \delta/n)$ to compare r to a random *competing* element $c \in S \setminus r$. If COMPARE returns 1 i.e., deems $\tilde{p}_{c,r} \leq 0$, it retains r as the running element and eliminates c

from S, but if COMPARE returns 2 i.e., deems $\tilde{p}_{c,r} \geq \varepsilon$, it eliminates r from S and updates c as the new running element.

Algorithm 11 SEQ-ELIMINATE

inputs
  Set S, bias ε, confidence δ
$n \leftarrow |S|$
$r \leftarrow$ a random $c \in S$, $S = S \setminus \{r\}$
while $S \neq \emptyset$ **do**
  Pick a random $c \in S$, $S = S \setminus \{c\}$.
  if COMPARE$(c, r, 0, \varepsilon, \frac{\delta}{n}) = 2$ **then**
    $r \leftarrow c$
  end if
end while
return r

We now bound the number of comparisons used by SEQ-ELIMINATE(S, ε, δ) and prove its correctness. Proof is in 3.A.3.

Theorem 28. SEQ-ELIMINATE(S, ε, δ) *uses* $O\left(\frac{|S|}{\varepsilon^2} \log \frac{|S|}{\delta}\right)$ *comparisons, and w.p.*$\geq 1 - \delta$ *outputs an* ε*-maximum.*

3.3.2 Reduction

Recall that, for $\delta \leq \frac{1}{n}$, SEQ-ELIMINATE is order-wise optimal. For $\delta \geq \frac{1}{n}$, here we present a reduction procedure that uses $O\left(\frac{n}{\varepsilon^2} \log \frac{1}{\delta}\right)$ comparisons and w.p.$\geq 1 - \delta$, outputs a subset S' of size $O(\sqrt{n \log n})$ and an element a such that either a is a $2\varepsilon/3$-maximum or S' contains an absolute maximum of S. Combining the reduction with SEQ-ELIMINATE results in an order-wise optimal algorithm.

We form the reduced subset S' by pruning S. We compare each element $e \in S$ with an anchor element a, test whether $\tilde{p}_{e,a} \leq 0$ or $\tilde{p}_{e,a} \geq 2\varepsilon/3$ using COMPARE, and retain all elements e for which COMPARE returns the second hypothesis. For S' to be of size $O(\sqrt{n \log n})$ we'd like to pick an anchor element that is among the top $O(\sqrt{n \log n})$ elements. But this can be

computationally hard and we show that it suffices to pick an anchor that is not $\varepsilon/3$-preferable to at most $O(\sqrt{n \log n})$ elements in S.

An element a is called an (ε, n')-*good anchor* if a is not ε-preferable to at most n' elements, i.e., $|\{e : e \in S \text{ and } \tilde{p}_{e,a} > \varepsilon\}| \leq n'$.

We now present the subroutine PICK-ANCHOR that finds a good anchor element.

Picking Anchor Element

PICK-ANCHOR$(S, n', \varepsilon, \delta)$ uses $O\left(\frac{n}{n'\varepsilon^2} \log \frac{1}{\delta} \log \frac{n}{n'\delta}\right)$ comparisons and w.p.$\geq 1-\delta$, outputs an (ε, n')-good anchor element. PICK-ANCHOR first picks randomly a set Q of $\frac{n}{n'} \log \frac{2}{\delta}$ elements from S without replacement. This ensures that w.p.$\geq 1-\delta$, Q contains at least one of the top n' elements. We then use SEQ-ELIMINATE to find an ε-maximum of Q.

Let the absolute maximum element of Q be denoted as q^*. Now an ε-maximum of Q is ε-preferable to q^*. Further, if Q contains an element in the top n' elements, there exists $n-n'$ elements worse than q^* in S. Thus by SST, the ε-maximum of Q is also ε-preferable to these $n-n'$ elements and hence the output of PICK-ANCHOR is an (ε, n')-good anchor element. PICK-ANCHOR is shown in appendix 3.A.4

We now bound the number of comparisons used by PICK-ANCHOR and prove its correctness. Proof is in 3.A.5.

Lemma 29. PICK-ANCHOR$(S, n', \varepsilon, \delta)$ *uses* $O\left(\frac{n}{n'\varepsilon^2} \log \frac{1}{\delta} \log \frac{n}{n'\delta}\right)$ *comparisons and w.p.*$\geq 1-\delta$*, outputs an* (ε, n')*-good anchor element.*

Remark 30. *Note that* PICK-ANCHOR$(S, cn, \varepsilon, \delta)$ *uses* $O_c\left(\frac{1}{\varepsilon^2}\left(\log \frac{1}{\delta}\right)^2\right)$ *comparisons where the constant depends only on c but not on n. Hence it is advantageous to use this method to pick near-maximum element when n is large.*

We now present PRUNE that takes an anchor element as input and prunes the set S using the anchor.

Pruning

Given an (ε_l, n')-good anchor element a, w.p.$\geq 1-\delta/2$, PRUNE$(S, a, n', \varepsilon_l, \varepsilon_u, \delta)$ outputs a subset S' of size $\leq 2n'$. Further, any element e that is at least ε_u-better than a i.e., $\tilde{p}_{e,a} \geq \varepsilon_u$ is in S' w.p.$\geq 1-\delta/2$.

PRUNE prunes S in multiple rounds. In each round t, for every element e in S, PRUNE tests whether $\tilde{p}_{e,a} \leq \varepsilon_l$ or $\tilde{p}_{e,a} \geq \varepsilon_u$ using COMPARE$(e, a, \varepsilon_l, \varepsilon_u, \delta/2^{t+1})$ and eliminates e if the first hypothesis i.e., $\tilde{p}_{e,a} \leq \varepsilon_l$ is returned. By Lemma 76, an element e that is ε_u better than a i.e., $\tilde{p}_{e,a} \geq \varepsilon_u$ passes the t^{th} round of pruning w.p.$\geq 1-\delta/2^{t+1}$. Thus by union bound, the probability that such an element is not present in the pruned set is $\leq \sum_{t=1}^{\infty} \delta/2^{t+1} \leq \delta/2$.

Now for element e that is not ε_l-better than a i.e., $\tilde{p}_{e,a} \leq \varepsilon_l$, by Lemma 76, the first hypothesis is returned w.p.$\geq 1-\delta/4$. Hence w.h.p., the number of such elements (not ε_l-better elements) is reduced by a factor of δ after each round. Since a is an (ε_l, n')-good anchor element, there are at most n' elements atleast ε_l-better than a. Thus the number of elements left in the pruned set after round t is at most $n' + n\delta^t$. Thus PRUNE succeeds eventually in reducing the size to $\leq 2n'$ (in $\leq \log_{1/\delta} \frac{n}{n'}$ rounds).

Algorithm 12 PRUNE

inputs
  Set S, element a, size n', lower bias ε_l, upper bias ε_u, confidence δ.
$t \leftarrow 1$
$S_1 \leftarrow S$
while $|S_t| > 2n'$ and $t < \log^2 n$ **do**
  Initialize: $Q_t \leftarrow \emptyset$
  for e in S_t **do**
    if COMPARE$(e, a, \varepsilon_l, \varepsilon_u, \delta/2^{t+1}) = 1$ **then**
      $Q_t \leftarrow Q_t \bigcup \{e\}$
    end if
  end for
  $S_{t+1} \leftarrow S_t \setminus Q_t$
  $t \leftarrow t+1$
end while
return S_t.

We now bound the number of comparisons used by PRUNE and prove its correctness. Proof is in 3.A.6.

Lemma 31. *If $n' \geq \sqrt{6n\log n}$, $\delta \geq \frac{1}{n}$ and a is an (ε_l, n')-good anchor element, then w.p.$\geq 1-\frac{\delta}{2}$, $\textsc{Prune}(S, a, n', \varepsilon_l, \varepsilon_u, \delta)$ uses $O\Big(\frac{n}{(\varepsilon_u-\varepsilon_l)^2}\log\frac{1}{\delta}\Big)$ comparisons and outputs a set of size less than $2n'$. Further if a is not an ε_u-maximum of S then w.p.$\geq 1-\frac{\delta}{2}$, the output set contains an absolute maximum element of S.*

3.3.3 Full Algorithm

We now present the main algorithm, OPT-MAXIMIZE that w.p.$\geq 1-\delta$, uses $O\Big(\frac{n}{\varepsilon^2}\log\frac{1}{\delta}\Big)$ comparisons and outputs an ε-maximum. For $\delta \leq \frac{1}{n}$, SEQ-ELIMINATE uses $O(\frac{n}{\varepsilon^2}\log\frac{1}{\delta})$ comparisons and hence we directly use SEQ-ELIMINATE. Below we assume $\delta > \frac{1}{n}$.

Here OPT-MAXIMIZE first finds an $(\varepsilon/3, \sqrt{6n\log n})$-good anchor element a using $\textsc{Pick-Anchor}(S, \sqrt{6n\log n}, \varepsilon/3, \frac{\delta}{4})$. Then using $\textsc{Prune}(S, a, \sqrt{6n\log n}, \varepsilon/3, 2\varepsilon/3, \frac{\delta}{4})$ with a, OPT-MAXIMIZE prunes S to a subset S' of size $\leq 2\sqrt{6n\log n}$ such that if a is not a $2\varepsilon/3$ maximum i.e. $\tilde{p}_{b^*,a} > 2\varepsilon/3$, S' contains the absolute maximum b^* w.p.$\geq 1-\delta/2$. OPT-MAXIMIZE then checks if a is a $2\varepsilon/3$ maximum by using $\textsc{Compare}(e, a, 2\varepsilon/3, \varepsilon, \delta/(4n))$ for every element $e \in S'$. If COMPARE returns first hypothesis for every $e \in S'$ then OPT-MAXIMIZE outputs a or else OPT-MAXIMIZE outputs $\textsc{Seq-Eliminate}(S', \varepsilon, \frac{\delta}{4})$.

Note that only one of these three cases is possible: (1) a is a $2\varepsilon/3$-maximum, (2) a is not an ε-maximum and (3) a is an ε-maximum but not a $2\varepsilon/3$-maximum. In case (1), since a is a $2\varepsilon/3$-maximum, by Lemma 76, w.p.$\geq 1-\delta/4$, COMPARE returns the first hypothesis for every $e \in S'$ and OPT-MAXIMIZE outputs a. In both cases (2) and (3), as stated above, w.p.$\geq 1-\delta/2$, S' contains the absolute maximum b^*. Now in case (2) since a is not an ε-maximum, by Lemma 76, w.p.$\geq 1-\delta/(4n)$, $\textsc{Compare}(b^*, a, 2\varepsilon/3, \varepsilon, \delta/(4n))$ returns the second hypothesis. Thus OPT-MAXIMIZE outputs $\textsc{Seq-Eliminate}(S', \varepsilon, \delta/4)$, which w.p.$\geq 1-\delta/4$, returns an ε-maximum of S' (recall that an ε-maximum of S' is an ε-maximum of S if S' contains b^*). Finally in case

(3), OPT-MAXIMIZE either outputs a or SEQ-ELIMINATE$(S', \varepsilon, \delta/4)$ and either output is an ε-maximum w.p.$\geq 1-\delta$. In the below Theorem, we bound comparisons used by OPT-MAXIMIZE

Algorithm 13 OPT-MAXIMIZE

inputs
  Set S, bias ε, confidence δ.
if $\delta \leq \frac{1}{n}$ **then**
  return SEQ-ELIMINATE(S, ε, δ)
end if
$a \leftarrow$ PICK-ANCHOR$(S, \sqrt{6n\log n}, \varepsilon/3, \frac{\delta}{4})$
$S' \leftarrow$ PRUNE$(S, a, \sqrt{6n\log n}, \varepsilon/3, 2\varepsilon/3, \frac{\delta}{4})$
for element e in S' **do**
  if COMPARE$(e, a, \frac{2\varepsilon}{3}, \varepsilon, \frac{\delta}{4n}) = 2$ **then**
    return SEQ-ELIMINATE$(S', \varepsilon, \frac{\delta}{4})$
  end if
end for
return a

and prove its correctness. Proof is in 3.A.7.

Theorem 32. *W.p.$\geq 1-\delta$,* OPT-MAXIMIZE(S, ε, δ) *uses* $O(\frac{n}{\varepsilon^2}\log\frac{1}{\delta})$ *comparisons and outputs an* ε*-maximum.*

3.4 Ranking

Recall that [FOPS17] considered a model with both SST and stochastic triangle inequality and derived an ε-ranking with $O\Big(\frac{n\log n(\log\log n)^3}{\varepsilon^2}\Big)$ comparisons for $\delta = \frac{1}{n}$. By constrast, we consider a more general model without stochastic triangle inequality and show that even a $1/4$-ranking with just SST takes $\Omega(n^2)$ comparisons for $\delta \leq \frac{1}{8}$.

To establish the lower bound, we reduce the problem of finding $1/4$-ranking to finding a coin with bias 1 among $\frac{n(n-1)}{2} - 1$ other fair coins. For this, we consider the following model with n elements $\{a_1, a_2, ..., a_n\}$: $\tilde{p}_{a_1,a_n} = \frac{1}{2}$, $\tilde{p}_{a_i,a_j} = \mu (0 < \mu < 1/n^{10})$, when $i < j$ and $(i,j) \neq (1,n)$. Note that this model satisfies SST but not stochastic triangle inequality. Also note that any

ranking where a_1 precedes a_n is an $1/4$-ranking and thus the algorithm only needs to order a_1 and a_n correctly. Now the output of a comparison between any two elements other than a_1 and a_n is essentially a fair coin toss (since μ is very small). Thus if we output a ranking without querying comparison between a_1 and a_n, then the ranking is correct w.p.$\approx \frac{1}{2}$ since a_1 and a_n must necessarily be ordered correctly. Now if an algorithm uses only $n^2/20$ comparisons then the probability that the algorithm queried at least one comparison between a_1 and a_n is less than $\frac{1}{2}$ and hence cannot achieve a confidence of $\frac{7}{8}$. Proof sketch in 3.B.1.

Theorem 33. *There exists a model that satisfies SST for which any algorithm requires* $\Omega(n^2)$ *comparisons to find a* $1/4$*-ranking with probability* $\geq 7/8$.

We also present a trivial ε-ranking algorithm in Appendix 3.B.2 that for any stochastic model with ranking (Weak Stochastic Transitivity), uses $O(\frac{n^2}{\varepsilon^2}\log\frac{n}{\delta})$ comparisons and outputs an ε-ranking w.p.$\geq 1-\delta$.

3.5 Borda Scores

We show that for general models, using $O(\frac{n}{\varepsilon^2}\log\frac{1}{\delta})$ comparisons w.p.$\geq 1-\delta$, we can find an ε-Borda maximum and using $O(\frac{n}{\varepsilon^2}\log\frac{n}{\delta})$ comparisons w.p.$\geq 1-\delta$, we can find an ε-Borda ranking.

Recall that Borda score $s(e)$ of an element e is the probability that e is preferable to an element picked randomly from S i.e., $s(e) = \frac{1}{n}\sum_{f\in S}\tilde{p}_{e,f}$. We first make a connection between Borda scores of elements and the traditional multi armed bandit setting. In the Bernoulli multi armed setting, every arm a is associated with a parameter $q(a)$ and pulling that arm results in a reward $\mathrm{B}(q(a))$, a Bernoulli random variable with parameter $q(a)$. Observe that we can simulate our pairwise comparisons setting as a traditional bandit arms setting by comparing an element with a random element where in our setting, for every element e, the associated parameter is $s(e)$. Thus PAC optimal algorithms derived under traditional bandit setting work for PAC Borda score

setting too. [ZC14] and several others derived a PAC maximum arm selection algorithms that use $O(\frac{n}{\varepsilon^2}\log\frac{1}{\delta})$ comparisons and find an arm with parameter at most ε less than the highest. This implies an ε-Borda maxing algorithm with the same complexity. Proof follows from reduction to Bernoulli multi-armed bandit setting.

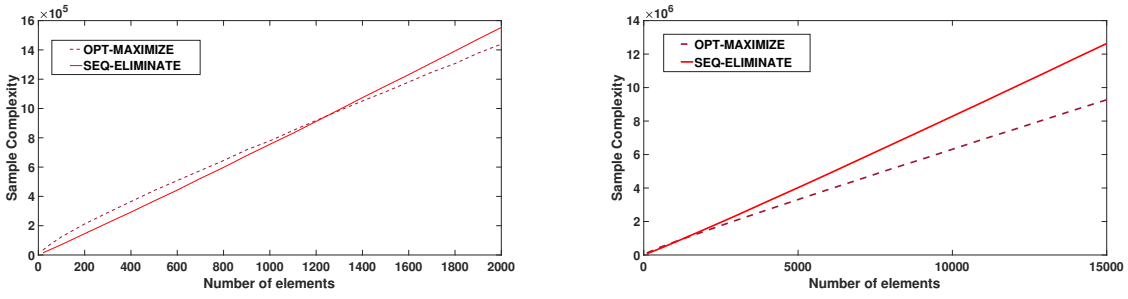
Theorem 34. *There exists an algorithm that uses $O(\frac{n}{\varepsilon^2}\log\frac{1}{\delta})$ comparisons and w.p.$\geq 1-\delta$, outputs an ε-Borda maximum.*

For ε-Borda ranking, we note that if we compare an element e with $\frac{2}{\varepsilon^2}\log\frac{2n}{\delta}$ random elements, w.p. $\geq 1-\delta/n$, the fraction of times e wins approximates the Borda score of e to an additive error of $\frac{\varepsilon}{2}$. Ranking based on these approximate scores results in an ε-Borda ranking. We present BORDA-RANKING in 3.C.1 that uses $\frac{2n}{\varepsilon^2}\log\frac{2n}{\delta}$ comparisons and w.p.$\geq 1-\delta$ outputs an ε-Borda ranking. Proof in 3.C.1.

Theorem 35. BORDA-RANKING(S,ε,δ) *uses $\frac{2n}{\varepsilon^2}\log\frac{2n}{\delta}$ comparisons and w.p.$\geq 1-\delta$ outputs an ε-Borda ranking.*

3.6 Experiments

In this section we validate the performance of our algorithms using simulated data. Since we essentially derived a negative result for ε-ranking, we consider only our ε-maxing algorithms - SEQ-ELIMINATE and OPT-MAXIMIZE for experiments. All results are averaged over 100 runs.



(a) small values of n (b) large values of n

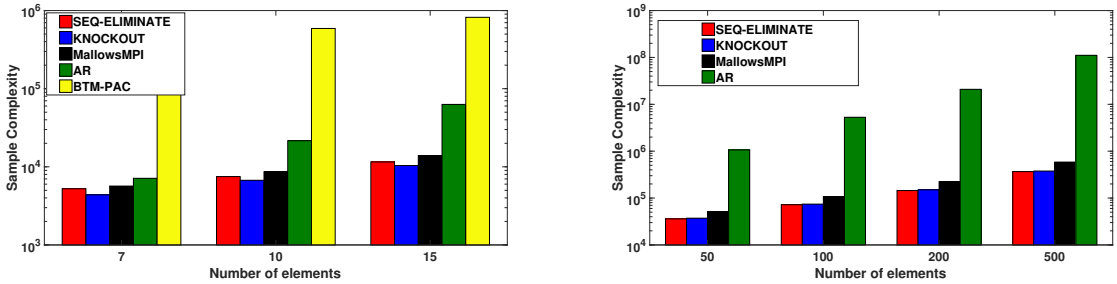
Figure 3.1: Comparison of SEQ-ELIMINATE and OPT-MAXIMIZE

Similar to [FOPS17, YJ11], we consider the stochastic model $p_{i,j} = 0.6\ \forall i < j$. We use maxing algorithms to find 0.05-maximum with error probability $\delta = 0.1$. Note that $i = 1$ is the unique 0.05-maximum under this model. In Figure 3.1, we compare the performance of SEQ-ELIMINATE and OPT-MAXIMIZE over different ranges of n. Figures2 3.1(a), 3.1(b) show that for small n i.e., $n \leq 1300$ SEQ-ELIMINATE performs well and for large n i.e., $n \geq 1300$, OPT-MAXIMIZE performs well. Since we are using $\delta = 0.1$, the experiment suggests that for $\delta \gtrapprox \frac{1}{n^{1/3}}$, OPT-MAXIMIZE uses fewer comparisons as compared to SEQ-ELIMINATE. Hence it would be beneficial to use SEQ-ELIMINATE for $\delta \leq \frac{1}{n^{1/3}}$ and OPT-MAXIMIZE for higher values of δ. In further experiments, we use $\delta = 0.1$ and $n < 1000$ so we use SEQ-ELIMINATE for better performance.

We compare SEQ-ELIMINATE with **BTM-PAC** [YJ11], **KNOCKOUT** [FOPS17], **MallowsMPI** [BFHS14], and **AR** [HSRW16] . **KNOCKOUT** and **BTM-PAC** are PAC maxing algorithms for models with SST and stochastic triangle inequality requirements. **AR** finds an element with maximum Borda score. **Mallows** finds the absolute best element under Weak Stochastic Transitivity.

We again consider the model: $p_{i,j} = 0.6\ \forall i < j$ and try to find a 0.05-maximum with error probability $\delta = 0.1$. Note that this model satisfies both SST and stochastic triangle inequality and under this model all these algorithms can find an ε-maximum. From Figure 3.2(a), we can see that **BTM-PAC** performs worse for even small values of n and from Figure 3.2(b), we can see that **AR** performs worse for higher values of n. One possible reason is that **BTM-PAC** is tailored for reducing regret in the bandit setting and in the case of **AR**, Borda scores of elements become approximately the same with increasing number of elements, leading to more comparisons. For this reason, we drop **BTM-PAC** and **AR** for further experiments.

We also tried **PLPAC** [SBFPH15] but it fails to achieve required accuracy of $1 - \delta$ since it is designed primarily for Plackett-Luce. For example, we considered the previous setting $p_{i,j} = 0.6\ \forall i < j$ with $n = 100$ and tried to find a 0.09-maximum with $\delta = 0.1$. Even though



(a) small values of n (b) large valuesof n

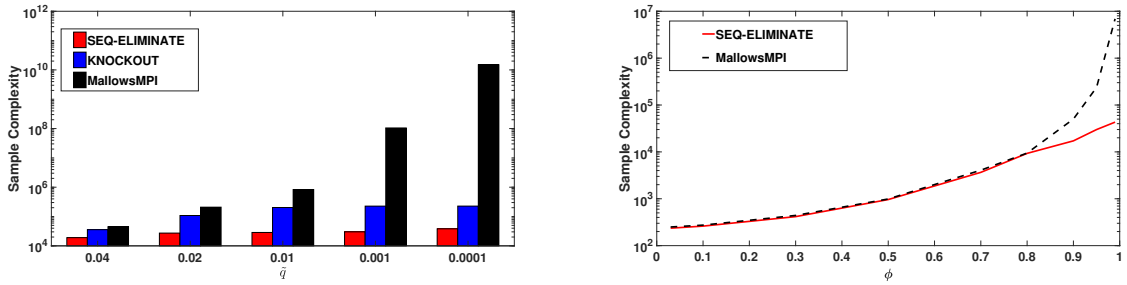
Figure 3.2: Comparison of Maxing Algorithms with Stochastic Triangle Inequality

PLPAC used almost same number of comparisons (57237) as SEQ-ELIMINATE (56683), PLPAC failed to find 0.09-maxima 20 out of 100 runs whereas SEQ-ELIMINATE found the maximum in all 100 runs.

In figure 3.3, we compare algorithms SEQ-ELIMINATE, **KNOCKOUT** [FOPS17] and **MallowsMPI** [BFHS14] for models that do not satisfy stochastic triangle inequality. In Figure 3.3(a), we consider the stochastic model $p_{1,j} = \frac{1}{2} + \tilde{q}\ \forall j \leq n/2,\ p_{1,j} = 1\ \forall j > n/2$ and $p_{i,j} = \frac{1}{2} + \tilde{q}\ \forall 1 < i < j$ where $\tilde{q} \leq 0.05$ and we pick $n = 10$. Observe that this model satisfies SST but not stochastic triangle inequality. Here again, we try to find a 0.05-maximum with $\delta = 0.1$. Note that any $i \leq n/2$ is a 0.05 maximum. From Figure 3.3(a), we can see that **MallowsMPI** uses more comparisons as $\tilde{q}$ decreases since **MallowsMPI** is not a PAC algorithm and tries to find the absolute maximum. Even though **KNOCKOUT** performs better than **MallowsMPI**, it fails to output a 0.05 maximum with probability 0.12 for $\tilde{q} = 0.001$ and 0.26 for $\tilde{q} = 0.0001$. Thus **KNOCKOUT** can fail when the model doesn't satisfy stochastic triangle inequality. We give an explanation for this behavior in Appendix 3.D. By constrast, even for $\tilde{q} = 0.0001$, SEQ-ELIMINATE outputted a 0.05 maximum in all runs and outputted the abosulte maximum in 76% of trials. We can also see that SEQ-ELIMINATE uses much fewer comparisons compared to the other two algorithms.

In Figure 3.3(b), we compare SEQ-ELIMINATE and **MallowsMPI** on the Mallows model, a model which doesn't satisfy stochastic triangle inequality. Mallows model can be specified with

one parameter ϕ. We consider $n = 10$ elements and find a 0.05-maximum with error probablility $\delta = 0.05$. From Figure 3.3(b) we can see that the performance of **MallowsMPI** gets worse as ϕ approaches 1, since comparison probabilities get close to $\frac{1}{2}$ whereas SEQ-ELIMINATE is not affected.



(a) No Triangle Inequality

(b) Mallows Model

Figure 3.3: Comparison of SEQ-ELIMINATE and MALLOWSMPI over Mallows Model

One more experiment is presented in Appendix 3.E.

3.7 Conclusion

We extended the study of PAC maxing and ranking to general models which satisfy SST but not stochastic triangle inequality. For PAC maxing, we derived an algorithm with linear complexity. For PAC ranking, we showed a negative result that any algorithm needs $\Omega(n^2)$ comparisons. We thus showed that removal of stochastic triangle inequality constraint does not affect PAC maxing but affects PAC ranking. We also ran experiments over simulated data and showed that our PAC maximum selection algorithms are better than other maximum selection algorithms.

For unconstrained models, we derived algorithms for PAC Borda maxing and PAC Borda ranking by making connections with traditional multi-armed bandit setting.

3.8 Acknowledgement

Chapter 3, in full, is a reprint of the material as it appears in *Advances in Neural Information Processing Systems*. Falahatgar, Moein, Yi Hao, Alon Orlitsky, Venkatadheeraj Pichapati, and Vaishakh Ravindrakumar, 2017. The dissertation author was the primary investigator and author of this paper.

3.A Maxing

3.A.1 COMPARE Algorithm

Motivated by a related algorithm in [FOPS17], we describe an adaptive version of COMPARE that stops when it is confident about the result, even if the number of comparisons is less than that specified in subsection 3.3.1. If $\tilde{p}_{i,j}$ is far outside $(\varepsilon_l, \varepsilon_u)$, this adaptive algorithm will terminate much sooner.

To do so, COMPARE maintains a varying confidence interval $\hat{c}$ such that w.p. $\geq 1-\delta$, $|\hat{\tilde{p}}_{i,j} - \tilde{p}_{i,j}| < \hat{c}$ after any number of comparisons. If at any time before the above number of comparisons, $|\hat{\tilde{p}}_{i,j} - \frac{\varepsilon_l+\varepsilon_u}{2}| > \hat{c}$, COMPARE simply returns the result based on the current $\hat{\tilde{p}}_{i,j}$.

Algorithm 14 COMPARE

inputs

element i, element j, bias lower limit $\varepsilon_l \geq 0$, bias upper limit $\varepsilon_u > \varepsilon_l$, confidence δ

initialize

$\varepsilon_m = (\varepsilon_l + \varepsilon_u)/2$, $\hat{\tilde{p}}_{i,j} \leftarrow 0$, $\hat{c} \leftarrow \frac{1}{2}$, $t \leftarrow 0$, $w \leftarrow 0$

while $|\hat{\tilde{p}}_{i,j} - \varepsilon_m| \leq \hat{c}$ and $t \leq \frac{2}{(\varepsilon_u - \varepsilon_l)^2} \log \frac{2}{\delta}$ **do**

Compare i and j

if i wins **then**

$w \leftarrow w + 1$

end if

$t \leftarrow t + 1$

$\hat{\tilde{p}}_{i,j} \leftarrow \frac{w}{t} - \frac{1}{2}$, $\hat{c} \leftarrow \sqrt{\frac{1}{2t} \log \frac{4t^2}{\delta}}$

end while

if $\hat{\tilde{p}}_{i,j} \leq \varepsilon_m$ **then**

return 1

end if

return 2

3.A.2 Proof of Lemma 76

We prove Lemma by dividing it into smaller parts. We first bound the comparisons used by COMPARE.

Lemma 36. *For* $\varepsilon_u > \varepsilon_l \geq 0$, COMPARE$(i, j, \varepsilon_l, \varepsilon_u, \delta)$ *uses* $\leq \frac{2}{(\varepsilon_u - \varepsilon_l)^2} \log \frac{2}{\delta}$ *comparisons.*

Proof. Notice that COMPARE$(i, j, \varepsilon_l, \varepsilon_u, \delta)$ compares elements i and j for at most $m = \frac{2}{(\varepsilon_u - \varepsilon_l)^2} \log \frac{2}{\delta}$ times and hence the Lemma follows. □

We show that under the first hypothesis namely $\tilde{p}_{i,j} \leq \varepsilon_l$, w.p.$\geq 1 - \delta$,

COMPARE$(i,j,\varepsilon_l,\varepsilon_u,\delta)$ returns 1.

Lemma 37. *For $\varepsilon_u > \varepsilon_l \geq 0$, if $\tilde{p}_{i,j} \leq \varepsilon_l$, then w.p.$\geq 1-\delta$,* COMPARE$(i,j,\varepsilon_l,\varepsilon_u,\delta)$ *outputs* 1.

Proof. Let $\hat{p}^t_{i,j}$ and $\hat{c}^t$ denote $\hat{p}_{i,j}$ and $\hat{c}$ respectively after t comparisons between i and j during COMPARE$(i,j,\varepsilon_l,\varepsilon_u,\delta)$. COMPARE$(i,j,\varepsilon_l,\varepsilon_u,\delta)$ outputs 2 only if $\hat{p}^t_{i,j} > \frac{1}{2} + \frac{\varepsilon_l+\varepsilon_u}{2} + \hat{c}^t$ for any $t < m = \frac{2}{(\varepsilon_l-\varepsilon_u)^2}\log\frac{2}{\delta}$ or if $\hat{p}^m_{i,j} > \frac{1}{2} + \frac{\varepsilon_l+\varepsilon_u}{2}$. We bound the probability of either of these events by $\frac{\delta}{2}$ and the result follows from the union bound.

By Hoeffding's inequality,

$$Pr\left(\hat{p}^t_{i,j} > \frac{1}{2} + \frac{\varepsilon_l+\varepsilon_u}{2} + \hat{c}^t\right) \leq Pr\left(\hat{p}^t_{i,j} > \frac{1}{2} + \varepsilon_l + \hat{c}^t\right) \leq e^{-2t(\hat{c}^t)^2} = e^{-\log\frac{4t^2}{\delta}} = \frac{\delta}{4t^2}.$$

By the union bound, $Pr\left(\exists t \text{ s.t. } \hat{p}^t_{i,j} > \frac{1}{2} + \frac{\varepsilon_l+\varepsilon_u}{2} + \hat{c}^t\right) \leq \sum_t \frac{\delta}{4t^2} \leq \frac{\delta}{2}$.

Similarly, by Hoeffding's inequality,

$$Pr\left(\hat{p}^m_{i,j} > \frac{1}{2} + \frac{\varepsilon_l+\varepsilon_u}{2}\right) \leq e^{-2m((\varepsilon_u-\varepsilon_l)/2)^2} = e^{-\log\frac{2}{\delta}} = \frac{\delta}{2}.$$ □

We now show that under the second hypothesis namely $\tilde{p}_{i,j} \geq \varepsilon_u$, w.p.$\geq 1-\delta$, COMPARE$(i,j,\varepsilon_l,\varepsilon_u,\delta)$ returns 2.

Lemma 38. *For $\varepsilon_u > \varepsilon_l \geq 0$, if $\tilde{p}_{i,j} \geq \varepsilon_u$, then w.p.$\geq 1-\delta$,* COMPARE$(i,j,\varepsilon_l,\varepsilon_u,\delta)$ *outputs* 2.

Proof. Let $\hat{p}^t_{i,j}$ and $\hat{c}^t$ denote $\hat{p}_{i,j}$ and $\hat{c}$ respectively after t comparisons between i and j during COMPARE$(i,j,\varepsilon_l,\varepsilon_u,\delta)$. COMPARE$(i,j,\varepsilon_l,\varepsilon_u,\delta)$ outputs 1 only if $\hat{p}^t_{i,j} < \frac{1}{2} + \frac{\varepsilon_l+\varepsilon_u}{2} - \hat{c}^t$ for any $t < m = \frac{2}{(\varepsilon_u-\varepsilon_l)^2}\log\frac{2}{\delta}$ or if $\hat{p}^m_{i,j} \leq \frac{1}{2} + \frac{\varepsilon_l+\varepsilon_u}{2}$. We bound the probability of either of these events by $\frac{\delta}{2}$ and the result follows from the union bound.

By Hoeffding's inequality,

$$Pr\left(\hat{p}^t_{i,j} < \frac{1}{2} + \frac{\varepsilon_l+\varepsilon_u}{2} - \hat{c}^t\right) \leq Pr\left(\hat{p}^t_{i,j} < \frac{1}{2} + \varepsilon_u - \hat{c}^t\right) \leq e^{-2t(\hat{c}^t)^2} = e^{-\log\frac{4t^2}{\delta}} = \frac{\delta}{4t^2}.$$

By the union bound, $Pr\Big(\exists t \text{ s.t. } \hat{p}^t_{i,j} < \frac{1}{2} + \frac{\varepsilon_l+\varepsilon_u}{2} - \hat{c}^t\Big) \leq \sum_t \frac{\delta}{4t^2} \leq \frac{\delta}{2}$.

Similarly, by Hoeffding's inequality,

$$Pr\left(\hat{p}^m_{i,j} \leq \frac{1}{2} + \frac{\varepsilon_l+\varepsilon_u}{2}\right) \leq e^{-2m((\varepsilon_u-\varepsilon_l)/2)^2} = e^{-\log\frac{2}{\delta}} = \frac{\delta}{2}. \qquad \square$$

Thus proof of Lemma 76 follows from Lemmas 36, 37 and 38.

3.A.3 Proof of Theorem 28

Proof. We first bound the total number of comparisons. Before each call of COMPARE (step 6), SEQ-ELIMINATE eliminates an element in step 5, hence COMPARE is called for exactly $n-1$ times. Further observe that COMPARE$(i,j,0,\varepsilon,\delta/n)$ always uses less than $\frac{2}{\varepsilon^2}\log\frac{2n}{\delta}$ comparisons. Hence the total comparisons used by SEQ-ELIMINATE(S,ε,δ) is

$$\leq \sum_{k=1}^{n-1} \frac{2}{\varepsilon^2}\log\frac{2n}{\delta} = O\Big(\frac{n}{\varepsilon^2}\log\frac{n}{\delta}\Big).$$

We now show that w.p.$\geq 1-\delta$, SEQ-ELIMINATE(S,ε,δ) outputs an ε-maximum. Let r^t, c^t denote the running and competing elements respectively before t^{th} run of COMPARE. Then by Lemmas 37 and 38, for any t, w.p. $\geq 1-\frac{\delta}{n}$,

$$\tilde{p}_{r^{t+1},r^t} \geq 0, \tag{3.1}$$

$$\tilde{p}_{r^{t+1},c^t} > -\varepsilon. \tag{3.2}$$

Further, by the union bound the probability that Equations 3.1 and 3.2 do not hold for some $1 \leq t \leq n$ is $\leq \delta$. Now let b^* be the absolute maximum element i.e., $\tilde{p}_{b^*,e} \geq 0\ \forall e \in S$. Then, either b^* is set as the running element before the first run of COMPARE i.e., $r^1 = b^*$ or b^* is the competing element at the t^{th} run of COMPARE for some $1 \leq t \leq n$ i.e., $c^t = b^*$. We show that in both cases, the output is an ε-maximum.

If $r^1 = b^*$, then by equation 3.1, future running elements are either b^* or better than b^*. Since b^* is the absolute maximum, future running elements must be b^* and hence b^* is the output.

If for some t, $c^t = b^*$, then by equation 3.2, $\tilde{p}_{r^{t+1},b^*} > -\varepsilon$. Further, by equation 3.1, $\tilde{p}_{r^l,r^{t+1}} \geq 0 \quad \forall l \geq t+1$. Hence by strong stochastic transitivity, $\tilde{p}_{r^n,b^*} > -\varepsilon$. Again, by strong stochastic transitivity, $\tilde{p}_{r^n,e} > -\varepsilon \; \forall e \in S$. Hence, the output is ε-maximum. □

3.A.4 PICK-ANCHOR algorithm

Algorithm 15 PICK-ANCHOR

1: **inputs**
2: Set S of size n, size n', bias ε, confidence δ.
3: Form a set Q by selecting $\min\left(\frac{n}{n'}\log\frac{2}{\delta}, n\right)$ random elements from S without replacement.
4: **return** SEQ-ELIMINATE$\left(Q, \varepsilon, \frac{\delta}{2}\right)$

3.A.5 Proof of Lemma 66

Proof. We first bound the number of comparisons used by PICK-ANCHOR$(S, n', \varepsilon, \delta)$. Since $|Q| \leq \frac{n}{n'}\log\frac{2}{\delta} = O(\frac{n}{n'}\log\frac{1}{\delta})$, Theorem 28 implies that the number of comparisons used by PICK-ANCHOR is

$$
\begin{aligned}
= \frac{2|Q|}{\varepsilon^2}\log\frac{2|Q|}{\delta} &= O\left(\frac{\frac{n}{n'}\log\frac{1}{\delta}}{\varepsilon^2}\log\frac{\frac{n}{n'}\log\frac{1}{\delta}}{\delta}\right) \\
&= O\left(\frac{n}{n'\varepsilon^2}\log\frac{1}{\delta}\left(\log\frac{n}{n'\delta} + \log\log\frac{1}{\delta}\right)\right) \\
&= O\left(\frac{n}{n'\varepsilon^2}\log\frac{1}{\delta}\log\frac{n}{n'\delta}\right).
\end{aligned}
$$

We show that w.p.$\geq 1-\delta$, the output element is an (ε, n')-good anchor element. We first show that Q contains atleast one of the top n' elements. We then show that the output element defeats one of the top n' elements with probability $\geq \frac{1}{2} - \varepsilon$. Hence by strong stochastic transitivity, the

output element defeats every element outside the top n' elements with probability $\geq \frac{1}{2} - \varepsilon$.

The probability that Q does not contain an element in top n' elements is $\leq \left(1 - \frac{n'}{n}\right)^{\frac{n}{n'}\log\frac{2}{\delta}} \leq \frac{\delta}{2}$. Note that the above statement is true even when size of Q is n. Let the best element in Q be denoted as q^*. By Theorem 28, w.p.$\geq 1-\delta/2$, the output element o of SEQ-ELIMINATE$(Q, \varepsilon, \frac{\delta}{2})$ is an ε-maximum of Q. Hence *w.p.* $\geq 1-\delta/2$, $\tilde{p}_{q^*,o} \leq \varepsilon$ and therefore by strong stochastic transitivity, for element e worse than q^*, $\tilde{p}_{e,o} \leq \tilde{p}_{q^*,o} \leq \varepsilon$. Since, w.p.$\geq 1-\delta/2$, the number of elements that are better than q^* is less than n', by the union bound, w.p.$\geq 1-\delta$, o is an (ε, n')-good anchor element. □

3.A.6 Proof of Lemma 31

We prove the Lemma by dividing it into three parts. We first show that an element e that is ε_u better than anchor a i.e., $\tilde{p}_{e,a} \geq \varepsilon_u$, is part of PRUNE$(S, a, n', \varepsilon_l, \varepsilon_u)$ w.p.$\geq 1-\delta/2$.

Lemma 39. *If $\tilde{p}_{e,a} \geq \varepsilon_u$, then w.p.$\geq 1-\delta/2$, the output set of* PRUNE$(S, a, n', \varepsilon_l, \varepsilon_u, \delta)$ *contains e.*

Proof. e is not part of output set only if $e \in Q_t$ for some t. Q_t will contain e only if S_t contains e and COMPARE$(e, a, \varepsilon_l, \varepsilon_u, \frac{\delta}{2^{t+1}})$ returns 1. By Lemma 38, since $\tilde{p}_{e,a} \geq \varepsilon_u$, probability that COMPARE$(e, a, \varepsilon_l, \varepsilon_u, \frac{\delta}{2^{t+1}})$ returns 1 is $\leq \frac{\delta}{2^{t+1}}$. Hence the probability that Q_t contains e is $\leq \frac{\delta}{2^{t+1}}$ and therefore by the union bound the probability that output set does not contain e is $\leq \sum_{t=1}^{\infty} \frac{\delta}{2^{t+1}} \leq \frac{\delta}{2}$. □

For $\frac{1}{n} \leq \delta \leq \frac{n'}{n}$, and if a is a good anchor element, we show that first round of pruning itself will reduce the set size to $2n'$ and hence bound the number of comparisons used by PRUNE.

Lemma 40. *If $n' \geq 8\log^2 n$, $\frac{1}{n} \leq \delta \leq \frac{n'}{n}$ and a is an (ε_l, n')-good anchor element then w.p.$\geq 1-\delta/2$,* PRUNE$(S, a, n', \varepsilon_l, \varepsilon_u, \delta)$ *uses $O\left(\frac{n}{(\varepsilon_u - \varepsilon_l)^2}\log\frac{1}{\delta}\right)$ comparisons and outputs a set of size at most $2n'$.*

Proof. If $n' \geq \frac{n}{2}$, the lemma is trivial. So let $n' < \frac{n}{2}$. Let the elements that defeat a with probability $\geq \frac{1}{2} + \varepsilon_l$ i.e., elements in set $\{e | \tilde{p}_{e,a} \geq \varepsilon_l\}$ be called good elements and the remaining elements be bad elements. Note that the number of bad elements in S_1 is $\geq n - n'$. We show that the number of bad elements in S_2 is $\leq n'$. An element e is part of S_2 only if COMPARE$(e, a, \varepsilon_l, \varepsilon_u, \delta/4)$ returns 2. By Lemma 37, each bad element in S_1 appears in S_2 w.p.$\leq \delta/4$. Therefore by the Chernoff bound, the probability that there are more than n' bad elements in S_2 is

$$\leq e^{-(n-n')D(\frac{n'}{n-n'}||\delta/4)} \leq e^{-\frac{n}{2}D(\frac{n'}{n}||\delta/4)} \leq e^{-\frac{n}{2}\frac{n'}{2n}} = e^{-n'/4} \leq \frac{1}{n^2} \leq \frac{\delta}{2}.$$

Since the number of good elements in S_1 is $\leq n'$, their size in S_2 is also $\leq n'$. Hence w.p.$\geq 1 - \delta/2$, $|S_2| \leq 2n'$ and therefore PRUNE stops after first iteration. Noting that PRUNE ran only for one iteration $t = 1$, COMPARE$(e, a, \varepsilon_l, \varepsilon_u, \delta/4)$ uses $O(\frac{1}{(\varepsilon_u - \varepsilon_l)^2} \log \frac{1}{\delta})$ comparisons. □

We now bound the number of comparisons used by PRUNE for higher values of δ by showing that after each round, the number of elements reduces roughly by a factor of δ.

Lemma 41. *If $n' > \sqrt{6n \log n}$, $\delta \geq \frac{n'}{n}$ and a is an (ε_l, n')-good anchor element, then w.p.$\geq 1 - \frac{\delta}{2}$,* PRUNE$(S, a, n', \varepsilon_l, \varepsilon_u, \delta)$ *uses $O\left(\frac{n}{(\varepsilon_u - \varepsilon_l)^2} \log \frac{1}{\delta}\right)$ comparisons and outputs a set of size less than $2n'$.*

Proof. As before let the elements that defeat a with probability $\geq \frac{1}{2} + \varepsilon_l$ i.e., elements in set $\{e | \tilde{p}_{e,a} \geq \varepsilon_l\}$ be called good elements and the remaining elements be bad elements. The number of good elements in S_1 is $\leq n'$ and number of bad elements in S_1 is $\geq n - n'$. We first show that in each iteration the number of bad elements decreases by atleast a factor of δ until it falls below n'. We then bound the number of rounds it takes for number of bad elements to fall below n'. Using this bound on number of rounds, we separately bound the number of comparisons used over bad and good elements.

Note that for every bad element e, COMPARE$(e, a, \varepsilon_l, \varepsilon_u, \delta')$ outputs 2 with probability $\leq \delta' \leq \delta/4$. Hence, if at the beginning of the round, the number of bad elements is more than n',

the probability that number of bad elements does not reduce by at least a factor of δ is

$$\leq e^{-n'D(\delta||\delta/4)} \leq e^{-n'\delta/2} \leq e^{-\frac{(n')^2}{2n}} \leq \frac{1}{n^3}$$

where the last inequality follows from $n' \geq \sqrt{6n\log n}$.

Now if the number of bad elements reduces by δ after each round, then the number of bad elements falls below n' in $t = 2\log_{\frac{1}{\delta}} \frac{n}{n'} \leq n$ rounds. Thus by the union bound, w.p.$\geq 1 - \frac{1}{n^2}$, the number of bad elements reduces by δ until the size becomes less than n'. Henceforth we assume this and bound the number of comparisons used.

We first bound the number of comparisons taken by PRUNE over bad elements. Number of bad elements in S_t is $\leq n\delta^{t-1}$. Since COMPARE$(e,a,\varepsilon_l,\varepsilon_u,\delta')$ uses $\frac{2}{(\varepsilon_u-\varepsilon_l)^2}\log\frac{1}{\delta'}$, the number of comparisons used by PRUNE over bad elements is

$$\begin{aligned}
&\leq \sum_{t=1}^{2\log_{1/\delta}\frac{n}{n'}} \frac{2n\delta^{t-1}}{(\varepsilon_u-\varepsilon_l)^2}\log\frac{2^{t+1}}{\delta} \\
&\leq \frac{2n}{(\varepsilon_u-\varepsilon_l)^2} \sum_{t=1}^{2\log_{1/\delta}\frac{n}{n'}} \left(\delta^{t-1}\log\frac{1}{\delta} + (t+1)(\delta)^{t-1}\log 2\right) \\
&= O\left(\frac{n}{(\varepsilon_u-\varepsilon_l)^2}\log\frac{1}{\delta}\right).
\end{aligned}$$

The last equality follows from the fact that if $\delta \leq 1/2$ (if $\delta > 1/2$, we can choose $\delta = 1/2$) then $\sum_t \delta^{t-1}$ and $\sum_t (t+1)\delta^{t-1}$ are bounded.

Now we bound the number of comparisons used by PRUNE over good elements. The

number of comparisons used by PRUNE over good elements is

$$\begin{aligned}
&\leq \sum_{t=1}^{2\log_{1/\delta}\frac{n}{n'}} \frac{n'}{(\varepsilon_u-\varepsilon_l)^2}\log\frac{2^{t+1}}{\delta} \\
&\leq \frac{n'}{(\varepsilon_u-\varepsilon_l)^2}\sum_{t=1}^{2\log_{1/\delta}\frac{n}{n'}}\left(\log\frac{1}{\delta}+(t+1)\log 2\right) \\
&\leq \frac{n'}{(\varepsilon_u-\varepsilon_l)^2}\left(\left(2\log_{1/\delta}\frac{n}{n'}\right)\log\frac{1}{\delta}+\left(2\log_{1/\delta}\frac{n}{n'}\right)^2\right) \\
&= O\left(\frac{n}{(\varepsilon_u-\varepsilon_l)^2}\log\frac{1}{\delta}\right).
\end{aligned}$$

□

Proof of Lemma 31 follows from Lemmas 39, 40 and 41.

3.A.7 Proof of Theorem 32

We prove the theorem by breaking it into parts. We first show that if anchor element a, the output of PICK-ANCHOR is a $2\varepsilon/3$-maximum then w.p.$\geq 1-\delta/4$, OPT-MAXIMIZE outputs a.

Lemma 42. *If a, the output of step 6 in* OPT-MAXIMIZE(S,ε,δ) *is a $\frac{2\varepsilon}{3}$-maximum of S, then w.p.$\geq 1-\frac{\delta}{4}$,* OPT-MAXIMIZE(S,ε,δ) *outputs a.*

Proof. a is not returned only if COMPARE in step 9 of OPT-MAXIMIZE returns 2. Since a is $\frac{2\varepsilon}{3}$-maximum of S, $\tilde{p}_{e,a}\leq\frac{2\varepsilon}{3}$, $\forall e\in S$. Then by Lemma 37, the probability that a single call of COMPARE$(e,a,2\varepsilon/3,\varepsilon,\frac{\delta}{4n})$ returns 2 is $\leq\frac{\delta}{4n}$. Hence by the union bound, the probability that COMPARE returns 1 for all calls in step 9 of OPT-MAXIMIZE is $\geq 1-\delta/4$. Therefore the Lemma follows. □

We now bound the number of comparisons used by OPT-MAXIMIZE in steps 1-6 and also prove some properties of PRUNE 's output set and anchor element.

Lemma 43. *For $\delta\geq\frac{1}{n}$, w.p.$\geq 1-\delta/2$, steps 1-6 in* OPT-MAXIMIZE(S,ε,δ) *uses $O\left(\frac{n}{\varepsilon^2}\log\frac{1}{\delta}\right)$ comparisons, outputs a set S' of size at most $\sqrt{24n\log n}$ and either a is a $2\varepsilon/3$-maximum element or S' contains the absolute maximum element.*

Proof. By Lemma 66, w.p.$\geq 1-\delta/4$, PICK-ANCHOR$(S, \sqrt{6n\log n}, \frac{\varepsilon}{3}, \frac{\delta}{4})$ uses $O(\frac{\sqrt{n\log n}}{\varepsilon^2}\log\frac{1}{\delta})$ comparisons and outputs an $(\varepsilon/3, \sqrt{6n\log n})$-good anchor element. From now we assume that a, the output of PICK-ANCHOR$(S, \sqrt{6n\log n}, \frac{\varepsilon}{3}, \frac{\delta}{4})$ is an $(\varepsilon/3, \sqrt{6n\log n})$-good achor element.

By Lemma 31, w.p.$\geq 1-\delta/4$, PRUNE$(S, a, \sqrt{6n\log n}, \varepsilon/3, 2\varepsilon/3, \delta/4)$ uses $O(\frac{n}{\varepsilon^2}\log\frac{1}{\delta})$ comparisons, outputs a set of size at most $\sqrt{24n\log n}$ and if a is not an $2\varepsilon/3$-maximum, then S' contains the absolute maximum.

And the Lemma follows by using the union bound. □

We now bound the number of comparisons used by OPT-MAXIMIZE during steps 8-13 assuming that either anchor element a is $2\varepsilon/3$-maximum or S' contains the absolute maximum of S.

Lemma 44. *For $\delta \geq \frac{1}{n}$, if a, the output of step 6 and S', the output of step 7 are such that either a is $2\varepsilon/3$-maximum of S or S' contains the absolute maximum element of S, then steps 8-13 of* OPT-MAXIMIZE(S, ε, δ) *uses $O(\frac{|S'|}{\varepsilon^2}\log\frac{n}{\delta})$ comparisons and w.p.$\geq 1-\delta/2$, outputs an ε-maximum.*

Proof. We first bound the number of comparisons. Each COMPARE$(e, a, 2\varepsilon/3, \varepsilon, \frac{\delta}{4n})$ uses $O(\frac{1}{\varepsilon^2}\log\frac{n}{\delta})$ comparisons and hence over all elements of S', COMPARE uses at most $O(\frac{|S'|}{\varepsilon^2}\log\frac{n}{\delta})$ comparisons. Further SEQ-ELIMINATE$(S', \varepsilon, \delta/4)$ uses $O(\frac{|S'|}{\varepsilon^2}\log\frac{n}{\delta})$ comparisons by Theorem 28.

If a is a $\frac{2\varepsilon}{3}$-maximum, then the result follows by Lemma 90.

Let a not be an $\frac{2\varepsilon}{3}$-maximum. Then S' contains the absolute maximum denoted here by b^*. Notice that by strong stochastic transitivity, an ε-maximum of S' is an ε-maximum of S since $b^* \in S'$. By Theorem 28, w.p.$\geq 1-\frac{\delta}{4}$, SEQ-ELIMINATE$(S', \varepsilon, \delta/4)$ outputs an ε-maximum. Now if $\tilde{p}_{b^*,a} > \varepsilon$, then w.p.$\geq 1-\frac{\delta}{4n}$, COMPARE$(b^*, a, 2\varepsilon/3, \varepsilon, \frac{\delta}{4n})$ returns 2 and hence a is not returned but SEQ-ELIMINATE$(S', \varepsilon, \delta/4)$ is returned. If $\tilde{p}_{b^*,a} \leq \varepsilon$, then a is an ε-maximum and hence returning a also results in an ε-maximum output. Lemma then follows by the union bound. □

Theorem 32 then follows from Theorem 28 and Lemmas 43 and 44.

3.B Ranking

3.B.1 Proof sketch for Theorem 33

Proof sketch. Consider the model where $\tilde{p}_{a_1,a_n} = 1/2$, $\tilde{p}_{a_i,a_j} = (0 <)\mu(\ll 1/n^{10})$, when $i < j$ and $(i,j) \neq (n,1)$. This model has an order: $a_1 > a_2 > \cdots > a_{n-1} > a_n$ i.e., $\tilde{p}_{a_i,a_j} > 0$ $\forall i < j$. Further this model satisfies strong stochastic transitivity since $\tilde{p}_{a_i,a_k} \geq \max(\tilde{p}_{a_i,a_j}, \tilde{p}_{a_j,a_k})$ $\forall i < j < k$.

We prove the Lemma by reducing the above model to the model where μ is replaced by 0. Note that new model does not satisfy strong stochastic transitivity but helps us in proving the Lemma.

Note that μ is so small that if we consider a model where we replace μ with 0, the comparisons behave essentially similarly. More formally, let model M_μ be the model we consider and M_0 be the model when μ is replaced with 0. Let S denote a sequence of comparisons where each element of the sequence includes the elements compared and its outcome. Further, for each sequence S, let $P_\mu(S)$ and $P_0(S)$ denote the probability of sequence S under models M_μ and M_0 respectively. Now consider a sequence S of comparisons of length $\leq n^2/20$. Then

$$\frac{P_0(S)}{P_\mu(S)} \geq \left(\frac{1/2}{1/2+\mu}\right)^{n^2/20} \geq e^{-n^2/(10n^{10})} \geq \frac{6}{7}$$

Thus the probability of any sequence of length $\leq \frac{n^2}{20}$ is approximately same under both models. Hence if there is an algorithm that uses $\frac{n^2}{20}$ comparisons and w.p.$\geq 7/8$ produces an $1/4$-ranking under M_μ model then applying same algorithm over M_0 model produces an $1/4$-ranking w.p.$\geq$ $\frac{7}{8} \cdot \frac{6}{7} = \frac{3}{4}$.

We now show that there exists no algorithm that uses $\frac{n^2}{20}$ comparisons and w.p.$\geq \frac{3}{4}$ generates a $1/4$-ranking under M_0, thus proving the Lemma. It is easy to see that any ordering outputted without querying the comparison between a_1 and a_n is a $1/4$-ranking w.p. exactly 1/2

since no order between a_1 and a_n can be deduced. Since the pair (a_1, a_n) is one random pair among $\binom{n}{2}$ pairs, the probability that the algorithm asks a comparison between this pair with $n^2/20$ comparisons is $< \frac{1}{2}$. So the probability that the output order contains a_1 and a_n in the right order is $< \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} = \frac{3}{4}$. □

3.B.2 Ranking Algorithm

We present STRONG-TRANSITIVITY-RANKING that uses $O(\frac{n^2}{\varepsilon^2} \log \frac{n}{\delta})$ comparisons and w.p.$\geq 1-\delta$ outputs an ε-ranking. STRONG-TRANSITIVITY-RANKING achieves this by approximating each $\tilde{p}_{i,j}$ with $\hat{p}_{i,j}$ to an additive error of $\frac{\varepsilon}{2}$. We first argue that there is an element e such that $\hat{p}_{e,j} \geq \frac{1}{2} - \varepsilon/2 \ \forall j \in S$ and such an element is an ε-maximum. Observe that if there is any element e such that $\hat{p}_{e,j} \geq \frac{1}{2} - \varepsilon/2 \ \forall j \in S$ then $p_{e,j} \geq \frac{1}{2} - \varepsilon \ \forall j \in S$ and hence e is an ε-maximum of S. Further recall that for the absolute maximum a^*, $\tilde{p}_{a^*,j} \geq \frac{1}{2} \ \forall j \in S$ and hence $\hat{p}_{a^*,j} \geq \frac{1}{2} - \varepsilon/2$ $\forall j \in S$. Therefore there will be at least one element e s.t. $\hat{p}_{e,j} \geq \frac{1}{2} - \varepsilon/2$ and such an element will be an ε-maximum of S. We find one such element, delete it from S and add it to the end of the ordered output set. We continue this process until we run out of elements in S. Since at every step we are adding an ε-maximum of the remaining set, the ordered output set will be an ε-ranking. We first present a subroutine ESTIMATE-PROBABILITY that compares two elements a and b for $\frac{1}{2\varepsilon^2} \log \frac{2}{\delta}$ and w.p.$\geq 1-\delta$ approximates $p(i,j)$ to an additive error of ε.

Algorithm 16 ESTIMATE-PROBABILITY

1: **inputs**
2: element i, element j, bias ε, confidence δ.
3: Compare i and j for $\frac{1}{2\varepsilon^2} \log \frac{2}{\delta}$ times.
4: **return** Fraction of times i won

Lemma 45. ESTIMATE-PROBABILITY$(i, j, \varepsilon, \delta)$ *uses* $\frac{1}{2\varepsilon^2} \log \frac{2}{\delta}$ *comparisons and w.p.*$\geq 1-\delta$ *approximates* $p_{i,j}$ *to an additive error of* ε.

Proof. Proof follows from Hoeffding's inequality. □

Algorithm 17 STRONG-TRANSITIVITY-RANKING

inputs
  Set S, bias ε, confidence δ
for every pair $\{i, j\}$ such that $i, j \in S$ **do**
  $\hat{p}_{i,j} \leftarrow$ ESTIMATE-PROBABILITY$(i, j, \varepsilon/2, \delta/n^2)$
  $\hat{p}_{j,i} \leftarrow 1 - p(i, j)$
end for
ordered set $T \leftarrow \emptyset$
while $|S| > 0$ **do**
  if $\exists e$ s.t. $\hat{p}_{e,f} \geq \frac{1}{2} - \varepsilon \; \forall f \in S$ **then**
    Add e at the end of T
    $S = S \setminus \{e\}$
  else
    Add S at the end of T
    return T
  end if
end while
return T

Lemma 46. STRONG-TRANSITIVITY-RANKING(S, ε, δ) *uses* $O(\frac{n^2}{\varepsilon^2} \log \frac{n}{\delta})$ *comparisons and w.p.*$\geq 1 - \delta$ *returns an* ε*-ranking.*

Proof. STRONG-TRANSITIVITY-RANKING calls ESTIMATE-PROBABILITY for $O(n^2)$ times, once for each pair and each $EP(i, j, \varepsilon/2, \delta/n^2)$ uses $O(\frac{1}{\varepsilon^2} \log \frac{n}{\delta})$ comparisons and hence bound on comparisons follow. W.p.$\geq 1 - \delta/n^2$, ESTIMATE-PROBABILITY$(i, j, \varepsilon/2, \delta/n^2)$ approximates $p_{i,j}$ with $\hat{p}_{i,j}$ such that $|p_{i,j} - \hat{p}_{i,j}| \leq \frac{\varepsilon}{2}$. By the union bound, w.p.$\geq 1 - \delta$, $|p_{i,j} - \hat{p}_{i,j}| \leq \frac{\varepsilon}{2} \; \forall i, j, \in S$. From here we assume that $|p_{i,j} - \hat{p}_{i,j}| \leq \frac{\varepsilon}{2} \; \forall i, j, \in S$ and show that the output is an ε-ranking. Let S^t denote the set of remaining elements in S after t elements are removed from S. We first

show that for $0 \leq t \leq n-1$, there is one element e such that $\hat{p}_{e,j} \geq \frac{1}{2} - \varepsilon \; \forall j \in S^t$ and such an element is an ε-maximum of S^t. Observe that if there is an element e such that $\hat{p}_{e,j} \geq \frac{1}{2} - \varepsilon/2$ $\forall j \in S^t$ then $p_{e,j} \geq \frac{1}{2} - \varepsilon \; \forall j \in S^t$ and hence e is an ε-maximum of S^t. Further recall that for the absolute maximum a^{t*} of S^t, $p_{a^{t*},j} \geq \frac{1}{2} \; \forall j \in S^t$ and hence $\hat{p}_{a^{t*},j} \geq \frac{1}{2} - \varepsilon/2 \; \forall j \in S^t$. Therefore there will be at least one element e s.t. $\hat{p}_{e,j} \geq \frac{1}{2} - \varepsilon/2$ and such an element will be an ε-maximum of S^t. STRONG-TRANSITIVITY-RANKING deletes one such element from S^t and adds it to the end of the ordered output set. Since for every t, STRONG-TRANSITIVITY-RANKING adds an ε-maximum of S^t to the output set, the Lemma follows. □

3.C Borda Scores

3.C.1 Ranking Algorithm for Borda Scores

Algorithm 18 ESTIMATE-BORDA-SCORE

inputs
  set S, element e, bias ε, confidence δ.
Initialize: $w \leftarrow 0$, $\hat{s} \leftarrow \frac{1}{2}$, $m \leftarrow \frac{1}{2\varepsilon^2} \log \frac{2}{\delta}$.
for $k = 1$ **to** $k = m$ **do**
  Compare e with random element $\in S$
  if e wins **then**
    $w \leftarrow w + 1$
  end if
  $\hat{s} = \frac{w}{k}$
end for
return $\hat{s}$

Lemma 47. ESTIMATE-BORDA-SCORE$(S, a, \varepsilon, \delta)$ *uses* $\frac{1}{2\varepsilon^2} \log \frac{2}{\delta}$ *comparisons and w.p.*$\geq 1 - \delta$

approximates $s(a)$ to an additive error of ε.

Proof. Proof follows from properties of ESTIMATE-BORDA-SCORE and Hoeffding's inequlity. □

Algorithm 19 BORDA-RANKING

inputs
  set S, bias ε, confidence δ.
Initialize: $b_e \leftarrow \frac{1}{2}$ for all $e \in S$
for element e in S **do**
  $b_e \leftarrow$ ESTIMATE-BORDA-SCORE$(S, e, \frac{\varepsilon}{2}, \frac{\delta}{n})$
end for
Rank S according to b_e.
return S.

Theorem 48. BORDA-RANKING(S, ε, δ) *uses* $\frac{2n}{\varepsilon^2} \log \frac{2n}{\delta}$ *comparisons and w.p.*$\geq 1 - \delta$ *outputs an* ε*-Borda ranking.*

Proof. BORDA-RANKING calls ESTIMATE-BORDA-SCORE for exactly n times and each call of ESTIMATE-BORDA-SCORE$(S, e, \varepsilon/2, \delta/n)$ uses $\frac{2}{\varepsilon^2} \log \frac{2n}{\delta}$ comparisons and hence the bound on comparisons follows.

Note that w.p.$\geq 1 - \delta/n$, ESTIMATE-BORDA-SCORE$(S, e, \varepsilon/2, \delta/n)$ approximates the Borda score of e to an additive error of $\varepsilon/2$. Let the approximate Borda score of element e be b_e. By the union bound, w.p.$\geq 1 - \delta$, BORDA-RANKING(S, ε, δ) approximates all Borda scores to an additive error of $\varepsilon/2$. From here, we assume that $|b_e - s(e)| \leq \frac{\varepsilon}{2}$ and show that ranking based on approximate Borda scores results in an ε-Borda ranking.

If an element e appears before element f in the output ranking then $b_e \geq b_f$. Since $|b_e - s(e)| \leq \frac{\varepsilon}{2}$ and $|b_f - s(f)| \leq \frac{\varepsilon}{2}$, $s(e) - s(f) = (b_e - b_f) + (s(e) - b_e) + (b_f - s(f)) \leq \varepsilon$. Hence the Lemma follows. □

3.D Why Knockout Fails

We will show that KNOCKOUT proposed in [FOPS17] fails under SST model without stochastic triangle inequality constraint.

Consider the model where $\tilde{p}_{a_1,a_j} = \mu \ \forall j < n/2$, $\tilde{p}_{a_1,a_j} = \frac{1}{2} \ \forall j \geq n/2$ and $\tilde{p}_{a_i,a_j} = \mu$ $\forall 1 < i < j$ for some $0 < \mu < \frac{1}{n^{10}}$. Observe that this model satisfies SST but not stochastic triangle inequality. Under this model, a_1 is the absolute maximum and any element in the set $\{a_i | i < n/2\}$ is a $1/4$-maximum. We show that under this model, w.p.$\geq 1/16$, KNOCKOUT$(S, 1/4, 1/16)$ fails to find a $1/4$-maximum.

KNOCKOUT pairs elements randomly in each round and compares each pair for a certain number of times and the winners proceed to the next round until there is only one element left. Observe that in the first round a_1 can get paired with an element from set $\{a_i | 1 < i < n/2\}$ w.p.$\approx 1/2$ and if that happens a_1 can lose the tie w.p.$\approx 1/2$. Hence a_1 can get eliminated in the first round w.p.$\approx 1/4$. Once a_1 is eliminated, in the second round, the elements will be approximately half from the first half of the original set and half from the second half. Since these elements are almost incomparable (comparisons between any two elements is now approximately a Bernoulli random variable with parameter $1/2$), each element is almost equally likely to be the final output. Therefore w.p.$\approx 1/8$, the output can be an element from second half of the set and hence not a $1/4$-maximum.

3.E Additional Experiment

To show why PAC maximum algorithms could be preferred to absolute maximum algorithms, once again we compare SEQ-ELIMINATE, **KNOCKOUT** and **MallowsMPI** for comparison probability values close to $1/2$. In Figure 3.4, we consider the stochastic model, $p_{1,j} = 0.6$ $\forall j > 1$ and $p_{i,j} = 0.5 + \tilde{q} \ \forall 1 < i < j$ where $\tilde{q} \ll 0.05$ with $n = 15$. Again, we find a 0.05-maximum with error probability $\delta = 0.1$. From Figure 3.4, we can observe that performance

of **MallowsMPI** gets much worse as $\tilde{q}$ decreases whereas SEQ-ELIMINATE and **KNOCKOUT** do not get affected since they are PAC maxing algorithms. Also observe that SEQ-ELIMINATE performs much better than **KNOCKOUT**.

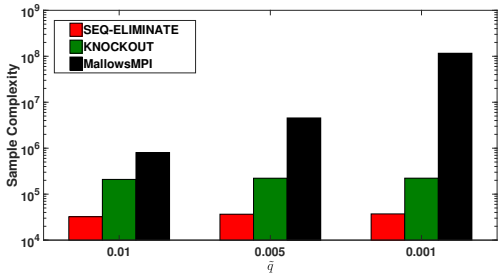


Figure 3.4: Comparison of Maximum Selection Algorithms for probability values close to $1/2$

Chapter 4

The Limits of Maxing, Ranking, and Preference Learning

4.1 Introduction

4.1.1 Background and motivation

Maximum selection (maxing) and sorting (ranking) are fundamental problems in Computer Science with numerous important applications. Deterministic versions of these problems are well studied.

In practical applications, comparisons are rarely deterministic. For example in soccer, when Real Madrid plays Barcelona the outcome is not always the same. Similarly, individual preferences in restaurants vary a lot. Other practical applications are in areas such as social choice [CN91, SCPX13], web search and information retrieval [RJ07, RKJ08], crowdsourcing [CBCTH13, gif], recommender systems [BMR10] and several others.

These practical applications and the intrinsic theoretical interest, has led to significant work on the probabilistic version of maxing and ranking. Yet the most general model for which maxing can be done using near-linear comparisons is not known. We consider the most general

transitive model that guarantees the existence of maximum and show that under this model any maxing algorithm requires quadratic many comparisons. We also consider a slightly more restrictive transitive model and propose a linear complexity maxing algorithm, making it the most general model known for which linear complexity maxing is possible. Also, for the most general known model with sub-quadratic complexity for ranking, we improve the complexity, making it orderwise optimal. We also propose an optimal algorithm that can simulate all pairwise comparisons.

4.1.2 Notation and problem formulation

Without loss of generality, let $[n] \stackrel{\text{def}}{=} \{1,2,...,n\}$ be the set of n elements. We consider probabilistic noisy comparisons i.e., whenever two elements i and j are compared, i is returned with an unknown probability $p_{i,j}$. There are no "ties" i.e., $p_{j,i} = 1 - p_{i,j}$. Let $\tilde{p}_{i,j} \stackrel{\text{def}}{=} p_{i,j} - \frac{1}{2}$ be the centered preference probability.

A maximal is an element i that is preferable to every other element i.e., $\tilde{p}_{i,j} \geq 0 \; \forall j$. A ranking is a permutation $\sigma_1, \sigma_2, ..., \sigma_n$ of $[n]$ such that $\tilde{p}_{\sigma_i,\sigma_j} \geq 0$ whenever $i > j$.

But sometimes maximal and ranking might not even exist. For example, consider the popular Rock-Paper-Scissor game i.e., $p_{1,2} = p_{2,3} = p_{3,1} = 1$. Notice that under this model there is neither a maximal nor a ranking. Hence we need additional constraints on pairwise probabilities $p_{i,j}$.

Notice that for ranking to exist, there must exist an ordering ($\succ$) among elements s.t. whenever $i \succ j$, $\tilde{p}_{i,j} \geq 0$. The models that have such an ordering are said to satisfy *Weak Stochastic Transitivity (WST)*. Observe that WST is sufficient for existence of both maximal and ranking.

More restrictive notions of transitivity are motivated and used in different contexts. *Strong Stochastic Transitivity (SST)* which assumes that whenever $i \succ j \succ k$, $\tilde{p}_{i,k} \geq \max(\tilde{p}_{i,j}, \tilde{p}_{j,k})$, as its name suggests is a stronger notion of transitivity that confines the model more than WST, hence less general. *Medium Stochastic Transitivity (MST)* [Sko10] sitting in between WST and SST,

assumes that whenever $i \succ j \succ k$, $\tilde{p}_{i,k} \geq \min(\tilde{p}_{i,j}, \tilde{p}_{j,k})$. From WST to MST to SST, the model becomes more restrictive.

Another model restriction used in some of the previous works *Stochastic Triangle Inequality (STI)*, assumes that whenever $i \succ j \succ k$, $\tilde{p}_{i,k} \leq \tilde{p}_{i,j} + \tilde{p}_{j,k}$. In this paper we propose maxing and ranking algorithms for models under various set of constraints.

There is also a concern with finding an exact maximal and ranking. Consider the case of $n = 2$ and $\tilde{p}_{1,2} \approx 0$. Notice that in this case where n is just 2, finding maximal and ranking could take arbitrarily many comparisons. Easy fix to alleviate this problem is to consider *Probably Approximately Correct (PAC)* formulation which we also adopt.

An element i is said to be ε-preferable to j if $\tilde{p}_{i,j} \geq -\varepsilon$. For $\varepsilon \in (0, 1/2)$, an ε-maximal is an element i that is ε-preferable to all elements i.e., $\tilde{p}_{i,j} \geq -\varepsilon \; \forall j$. Given $0 < \varepsilon < 1/2, 0 < \delta \leq 1/2$, a PAC maxing algorithm must output an ε-maximal with probability $\geq 1 - \delta$. Similarly, an ε-ranking is a permutation $\sigma_1, \sigma_2, ..., \sigma_n$ of $[n]$ such that σ_i is ε-preferable to σ_j whenever $i > j$. Given $0 < \varepsilon < 1/2$, $0 < \delta \leq 1/2$, a PAC ranking algorithm must output an ε-ranking with probability $\geq 1 - \delta$.

4.1.3 Related work

Researchers initially considered more restrictive models. [FRPU94] considered constant noise model i.e., $\tilde{p}_{i,j} = \alpha > 0$ if $i \succ j$ and presented a maxing algorithm that uses $O\left(\frac{n}{\alpha^2} \log \frac{1}{\delta}\right)$ comparisons and outputs maximal with probability $\geq 1 - \delta$. It also presented a ranking algorithm that uses $O\left(\frac{n \log n}{\alpha^2}\right)$ comparisons and outputs ranking with probability $\geq 1 - 1/n$.

Another set of widely-studied restrictive models are parametric ones. [SBFPH15] considered one of the most popular parametric models, Plackett-Luce [Pla75, Luc05] and presented PAC maxing and ranking algorithms that use $O\left(\frac{n}{\varepsilon^2} \log \frac{n}{\varepsilon\delta}\right)$ and $O\left(\frac{n \log n}{\varepsilon^2} \log \frac{n}{\varepsilon\delta}\right)$ comparisons respectively.

Researchers also considered models that are more general than parametric models, yet still

more restrictive than WST. [YJ11] considered models that satisfy both SST and STI and derived a PAC maxing algorithm that uses $O\left(\frac{n}{\varepsilon^2}\log\frac{n}{\varepsilon\delta}\right)$ comparisons. Later [FOPS17] considered same model and proposed an optimal PAC maxing algorithm that uses $O\left(\frac{n}{\varepsilon^2}\log\frac{1}{\delta}\right)$ comparisons. It also proposed a PAC ranking algorithm that with probability $\geq 1-1/n$, outputs an ε-ranking using $O\left(\frac{n\log n(\log\log n)^3}{\varepsilon^2}\right)$ comparisons, $(\log\log n)^3$ times the known lower bound. Until now, it was not known if the additional $(\log\log n)^3$ factor is necessary for PAC ranking.

[FHO$^+$17] considered models that satisfy only SST but not necessarily STI and proposed an optimal PAC maxing algorithm that uses $O\left(\frac{n}{\varepsilon^2}\log\frac{1}{\delta}\right)$ comparisons. They also showed that there exists a model which satisfies SST and yet no algorithm can find an ε-ranking for this model using $o(n^2)$ comparisons, establishing a lower bound of $\Omega(n^2)$ comparisons once STI property is dropped.

Among other related works we can point out [BFSH14, LGAL14, DHS$^+$15, HFCB08], who considered models more general than WST under different definitions of maximum and ranking. More discussion about these models can be found in Appendix 4.G. [BFHS14, MSE17] considered the non-PAC version and [RA14, NOS12, NOS16, JKSO16] considered the non-adaptive version of this problem. Also [AFJ$^+$16, AFHN15] considered the deterministic adversarial version of maxing and ranking. [SBW16, C$^+$15, SBGW16] studied the problem of estimating pairwise probabilities in non-adaptive setting.

4.2 New results and Outline

Maxing Linear-complexity maxing algorithm under SST by [FHO$^+$17] encourages the search for a linear-complexity maxing algorithm for models with only WST properties. Two questions then arise: 1a) Is a linear complexity PAC maxing algorithm possible for models with only WST property? 1b) If not, does there exist a model more general than SST and less general than WST for which a linear complexity PAC maxing is possible?

We resolve both questions in this paper: 1a) No. Theorem 49 in Section 4.3 shows that there are WST models for which any PAC maxing algorithm requires $\Omega(n^2)$ comparisons. 1b) Yes. In Theorem 56 in Section 4.4, we derive a PAC maxing algorithm for MST model that uses $O\left(\frac{n}{\varepsilon^2}\log\frac{1}{\delta}\right)$ comparisons for $\delta \geq \min(1/n, e^{-n^{1/4}})$.

Ranking Motivated by the previous results of ranking under SST + STI, three questions arise: 2a) For models with SST + STI, is the additional $(\log\log n)^3$ factor necessary for PAC ranking algorithms? 2b) Since the near-linear complexity of ranking under SST + STI changes to quadratic complexity by dropping STI [FHO$^+$17], is there a sub-quadratic algorithm for ranking under MST + STI? 2c) For models with SST + STI, since PAC ranking is possible with near linear complexity, is it also possible to approximate all pairwise probabilities to accuracy of ε using near linear number of comparisons?

We essentially resolve all three questions. 2a) No. In Theorem 57 in Section 4.5, we improve the PAC ranking algorithm for models with SST + STI removing additional $(\log\log n)^3$ factor and hence making it optimal. 2b) No. Theorem 58 in Section 4.6 shows that there is a model with MST+STI, for which any PAC ranking algorithm requires $\Omega(n^2)$ comparisons. 2c) Yes. For models with SST + STI, in Theorems 59 and 60 in Sections 4.7, we present an optimal algorithm that uses $O\left(\frac{n\min(n,1/\varepsilon)\log n}{\varepsilon^2}\right)$ comparisons and approximates all pairwise probabilities to accuracy of ε with probability $\geq 1 - 1/n$.

We present experiments over simulated data in Section 4.8 and end with our conclusions in Section 5.6.

Interpretation Table 4.1 summarizes all known results for problems of maxing, ranking, and finding pairwise probabilities under different transitive properties. Notice that under the most general model WST, all these problems require quadratic many comparisons and under the most restrictive model SST + STI, all problems have optimal algorithms with near-linear complexity. For MST and WST models adding STI property does not influence complexity for any problem. But for SST model adding STI property facilitates near-linear complexity algorithms for PAC

Table 4.1: Comprehensive results for maxing, ranking and finding $p_{i,j}$

$*$: for $\delta \geq \frac{1}{n}$, $**$: for $\delta \geq \min(1/n, e^{-n^{1/4}})$

Model	Maxing	Ranking	Finding $p_{i,j}$
SST with STI	$\Theta\left(\frac{n}{\varepsilon^2}\log\frac{1}{\delta}\right)$ [FOPS17]	$\Theta\left(\frac{n\log n}{\varepsilon^2}\right)^*$ Section 4.5	$\Theta\left(\frac{n\min(n,1/\varepsilon)\log n}{\varepsilon^2}\right)^*$ Section 4.7
SST	$\Theta\left(\frac{n}{\varepsilon^2}\log\frac{1}{\delta}\right)$ [FHO+17]	$\Omega(n^2)$ [FHO+17]	$\Omega(n^2)$
MST with STI and MST	$\Theta\left(\frac{n}{\varepsilon^2}\log\frac{1}{\delta}\right)^{**}$ Section 4.4	$\Omega(n^2)$ Section 4.6	$\Omega(n^2)$
WST with STI and WST	$\Omega(n^2)$ Section 4.3	$\Omega(n^2)$ Section 4.6	$\Omega(n^2)$

ranking and approximating pairwise probabilities.

It is easy to see that once all pairwise probabilities are approximated to accuracy of $\varepsilon/2$, one can find an ε-maximum and an ε-ranking. Hence approximating pairwise probabilities is harder than PAC ranking and lower bound for PAC ranking implies a lower bound for problem of approximating pairwise probabilities. Therefore in Table 4.1 lower bounds for finding p_{ij} follow from lower bounds for ranking. Further in Appendix 4.B.1, under WST model, we present a trivial algorithm that with probability $\geq 1-\delta$, estimates all pairwise probabilities to accuracy of ε using $O\left(\frac{n^2}{\varepsilon^2}\log\frac{n}{\delta}\right)$ comparisons. Hence upper bound of $O\left(\frac{n^2}{\varepsilon^2}\log\frac{n}{\delta}\right)$ follows for all problems.

4.3 PAC maxing for WST

We show the lower bound of $\Omega(n^2)$ for maxing under WST by presenting an example for which any algorithm requires $\Omega(n^2)$ comparisons to output a $1/4$-maximum for $\delta \leq 1/8$.

To establish the lower bound, we reduce the problem of finding a $1/4$-maximum to finding the left most piece of a linear jigsaw puzzle. We consider the following model with n elements $S = \{a_1, a_2, \ldots, a_n\}$: $\tilde{p}_{a_i,a_{i+1}} = \frac{1}{2} \forall i < n$, and $\tilde{p}_{a_i,a_j} = \mu(0 < \mu < 1/n^{10}), \forall j > i+1$. This model

satisfies WST since there exists an underlying order $\succ$, $a_i \succ a_j$ if $i < j$ (because $\tilde{p}_{a_i,a_j} > 0$) and a_1 is the only $1/4$-maximum under this model.

Observe that a_i is always preferred to a_{i+1}, but for every non consecutive pair, comparison output is almost a fair coin flip. We make the problem simpler by giving the extra information of whether two non consecutive elements are being compared. Notice that this only makes the problem easier, namely, complexity for modified problem is smaller than that of original problem.

The modified problem is similar to a linear jigsaw puzzle where if we compare two pieces we will know if pieces are adjacent or not and if adjacent, which piece is on the left, the goal is to find the left most piece. We show that w.h.p., any algorithm neither finds more than $n/32$ connections (a set of neighbors) nor asks $\Omega(n)$ comparisons for the left most piece. We use this to show the lower bound. The proof is in Appendix 4.A.

Theorem 49. *There exists a model that satisfies WST for which any algorithm requires* $\Omega(n^2)$ *comparisons to find a* $1/4$*-maximum with probability* $\geq 7/8$.

4.4 PAC maxing for MST

Outline In this section, we propose OPT-MAX, a linear complexity maxing algorithm for MST. In the process, we present two other suboptimal maxing algorithms SOFT-SEQ-ELIM, NEAR-OPT-MAX and use them as building blocks in OPT-MAX. SOFT-SEQ-ELIM finds an ε-maximum with quadratic complexity. Its performance depends on the starting element (anchor). NEAR-OPT-MAX first finds a good anchor and then uses SOFT-SEQ-ELIM, guaranteeing near linear comparison complexity. OPT-MAX builds on NEAR-OPT-MAX and finds an ε-maximum in linear-complexity for $\delta \geq \min(1/n, e^{-n^{1/4}})$.

4.4.1 SOFT-SEQ-ELIM

Before presenting SOFT-SEQ-ELIM, we first present the subroutine COMPARE we use to compare two elements.

COMPARE COMPARE takes 5 parameters : two elements i, j that need to be compared, lower bias ε_l, upper bias ε_u, confidence δ and deems if $\tilde{p}_{i,j} < \varepsilon_l$ or $\tilde{p}_{i,j} > \varepsilon_u$. It compares i and j for $\frac{8}{(\varepsilon_u-\varepsilon_l)^2}\log\frac{2}{\delta}$ times. Let $\hat{p}_{i,j}$ be the fraction of times i won and $\hat{\tilde{p}}_{i,j} = \hat{p}_{i,j} - 1/2$. If $\hat{\tilde{p}}_{i,j} < \frac{3\varepsilon_l}{4} + \frac{\varepsilon_u}{4}$, then COMPARE deems $\tilde{p}_{i,j} < \varepsilon_l$ (returns 1), if $\hat{\tilde{p}}_{i,j} > \frac{\varepsilon_l}{4} + \frac{3\varepsilon_u}{4}$, then COMPARE deems $\tilde{p}_{i,j} > \varepsilon_u$ (returns 3) and for other ranges of $\hat{\tilde{p}}_{i,j}$, COMPARE not able to take a decision, returns 2.

Lemma 50 bounds comparisons used by COMPARE and proves its correctness. COMPARE and its analysis is presented in Appendix 4.C.2.

Lemma 50. *For $\varepsilon_u > \varepsilon_l$,* COMPARE$(i, j, \varepsilon_l, \varepsilon_u, \delta)$ *uses $\leq \frac{8}{(\varepsilon_u-\varepsilon_l)^2}\log\frac{2}{\delta}$ comparisons and if $\tilde{p}_{i,j} < \varepsilon_l$, then w.p.$\geq 1-\delta$, it returns* 1*, else if $\tilde{p}_{i,j} > \varepsilon_u$, w.p.$\geq 1-\delta$, it returns* 3*. Further if $\tilde{p}_{i,j} \leq (\varepsilon_l+\varepsilon_u)/2$, w.p.$\geq 1-\delta$, it does not return* 3 *and if $\tilde{p}_{i,j} > (\varepsilon_l+\varepsilon_u)/2$, w.p.$\geq 1-\delta$, it does not return* 1*.*

SOFT-SEQ-ELIM SOFT-SEQ-ELIM takes 5 parameters: input set S, starting anchor element r, lower bias ε_l, upper bias ε_u and confidence δ. SOFT-SEQ-ELIM happens in rounds. In each round, it compares the current anchor a with remaining elements one by one using COMPARE. Due to probabilistic nature, we cannot exactly compare if $\tilde{p}_{e,a} > \varepsilon_u$ vs $\tilde{p}_{e,a} \leq \varepsilon_u$. Hence we compare if $\tilde{p}_{e,a} > \varepsilon_u$ vs $\tilde{p}_{e,a} < \varepsilon_l$. For an element e, if COMPARE deems $\tilde{p}_{e,a} < \varepsilon_l$, then SOFT-SEQ-ELIM eliminates that element and if COMPARE deems $\tilde{p}_{e,a} > \varepsilon_u$, then SOFT-SEQ-ELIM updates current anchor element to e and eliminates a. This process is continued until the current anchor element is not updated after comparing with all remaining elements and then SOFT-SEQ-ELIM outputs final anchor element.

If $\tilde{p}_{e,a} < \varepsilon_l$ or $\tilde{p}_{e,a} > \varepsilon_u$, COMPARE deems correctly. If $\varepsilon_l \leq \tilde{p}_{e,a} \leq \varepsilon_u$, then COMPARE can sometimes fail to output any decision and in that case, SOFT-SEQ-ELIM neither eliminates

that element nor updates the anchor element, it just moves to next remaining element in S.

Algorithm 20 SOFT-SEQ-ELIM

inputs
  Set S, element r, lower bias ε_l, upper bias ε_u, confidence δ
$Q = S \setminus \{r\}$
while $Q \neq \emptyset$ **do**
  $r' = r$, $Q' = \emptyset$
  for $c \in Q$ **do**
    $k = \text{COMPARE}(c, r, \varepsilon_l, \varepsilon_u, \frac{2\delta}{|S|^2})$
    if $k == 1$ **then**
      $Q' = Q' \bigcup \{c\}$.
    else if $k == 3$ **then**
      $r \leftarrow c$
      $Q' = Q' \bigcup \{c\}$
      break
    end if
  end for
  if $r == r'$ **then**
    break
  end if
  $Q = Q \setminus Q'$
end while
return r

Theoretically, performance of SOFT-SEQ-ELIM strongly depends on the starting anchor element r. To define a good anchor element, similar to [FHO+17], an element a is called an (ε, m)-*good anchor* if a is not ε-preferable to at most m elements, i.e., $|\{e : e \in S \text{ and } \tilde{p}_{e,a} > \varepsilon\}| \leq m$. We show that every element for which initial anchor r is ε_l-preferable is deemed bad and gets eliminated after its first comparison round and hence comparisons spent on all such elements is $O(|S|)$. Since initial anchor r is an (ε_l, m)-*good anchor* element, there are only m elements for which r is not ε_l-preferable. We later show that only these elements can become anchors, leading to at most m changes of anchors. Therefore each such element gets compared in at most m rounds and hence we can bound total comparison rounds by $O(|S| + m^2)$. Lemma 51 bounds comparisons used by SOFT-SEQ-ELIM and proves its correctness. Proof is in Appendix 4.C.3.

Lemma 51. *If r is an (ε_l, m)-good anchor element, w.p.$\geq 1-\delta$,* SOFT-SEQ-ELIM$(S, r, \varepsilon_l, \varepsilon_u, \delta)$ *uses $O\left(\frac{|S|+m^2}{(\varepsilon_u-\varepsilon_l)^2}\log\frac{|S|}{\delta}\right)$ comparisons and outputs $\hat{r}$, an ε_u maximum of S, such that either $\hat{r}=r$ or $\tilde{p}_{\hat{r},r} > \frac{\varepsilon_l+\varepsilon_u}{2}$.*

Corollary 52 bounds comparisons used by SOFT-SEQ-ELIM for any starting anchor. Proof follows from Lemma 51

Corollary 52. *For any r, w.p.$\geq 1-\delta$,* SOFT-SEQ-ELIM$(S, r, \varepsilon_l, \varepsilon_u, \delta)$ *uses $O\left(\frac{|S|^2}{(\varepsilon_u-\varepsilon_l)^2}\log\frac{|S|}{\delta}\right)$ comparisons and outputs $\hat{r}$, an ε_u maximum of S, such that either $\hat{r}=r$ or $\tilde{p}_{\hat{r},r} > \frac{\varepsilon_l+\varepsilon_u}{2}$.*

Now we build on SOFT-SEQ-ELIM and propose a near linear algorithm NEAR-OPT-MAX.

4.4.2 NEAR-OPT-MAX

NEAR-OPT-MAX(S, ε, δ) w.p.$\geq 1-\delta$, uses $O\left(\frac{|S|}{\varepsilon^2}\left(\log\frac{|S|}{\delta}\right)^2\right)$ comparisons and outputs an ε-maximum of S.

Since complexity of SOFT-SEQ-ELIM depends on the initial anchor element, if we can pick a good initial anchor element, then we can reduce the number of comparisons. One way to pick a good initial anchor element is to find an $\varepsilon/2$-maximum of a randomly picked subset.

Lemma 90 shows that an ε-maximum of a randomly picked subset is a good anchor element. Proof in Appendix 4.C.4.

Lemma 53. *If r is an ε-maximum of a set Q, formed by picking m elements randomly from S, then w.p.$\geq 1-\delta$, r is an $\left(\varepsilon, \frac{|S|}{m}\log\frac{|S|}{\delta}\right)$-good anchor element of S.*

NEAR-OPT-MAX(S, ε, δ) first picks a random subset Q of size $\sqrt{|S|\log\frac{4|S|}{\delta}}$ and uses SOFT-SEQ-ELIM to find an $\varepsilon/2$-maximum of Q.

By Lemma 90, w.p.$\geq 1-\delta/4$, an $\varepsilon/2$-maximum of Q will be an $(\varepsilon/2, \sqrt{|S|\log\frac{4|S|}{\delta}})$-*good anchor* element. NEAR-OPT-MAX then uses SOFT-SEQ-ELIM with $\varepsilon/2$-maximum of Q as initial anchor to find an ε-maximum of S. Since the initial anchor is provably good, we are able to bound the comparisons.

Algorithm 21 NEAR-OPT-MAX

inputs
  Set S, bias ε, confidence δ
Form a set Q by selecting $\sqrt{|S|\log\frac{4|S|}{\delta}}$ random elements from S without replacement.
$a \leftarrow$ random element from Q, $Q = Q \setminus \{a\}$
$r \leftarrow$ SOFT-SEQ-ELIM$\left(Q, a, 0, \frac{\varepsilon}{2}, \frac{\delta}{4}\right)$, $S = S \setminus \{r\}$
return SOFT-SEQ-ELIM$(S, r, \varepsilon/2, \varepsilon, \delta/2)$

Lemma 54 bounds the comparisons used by NEAR-OPT-MAX and proves its correctness.

Lemma 54. *With probability* $\geq 1-\delta$, NEAR-OPT-MAX(S,ε,δ) *uses* $O\left(\frac{|S|}{\varepsilon^2}\left(\log\frac{|S|}{\delta}\right)^2\right)$ *comparisons and outputs an* ε*-maximum of* S.

We build on NEAR-OPT-MAX and derive an optimal algorithm for $\delta \geq \min(1/|S|, e^{-|S|^{1/4}})$.

4.4.3 Optimal linear Algorithm

We first present an algorithm that is optimal for low ranges of δ i.e., $\min(e^{-|S|^{1/4}}, 1/|S|) \leq \delta \leq \frac{1}{|S|^{1/3}}$.

Low ranges of δ

We first find a good anchor, this time using NEAR-OPT-MAX and then use SOFT-SEQ-ELIM with NEAR-OPT-MAX output as initial anchor.

OPT-MAX-LOW picks a random subset of size $|S|^{3/4}$ and finds an $\varepsilon/2$-maximum of this set using NEAR-OPT-MAX. We later show that output is an $(\varepsilon/2, O(\sqrt{|S|}))$-*good anchor* element of S. OPT-MAX-LOW then uses SOFT-SEQ-ELIM with the previous output as initial anchor to find an ε-maximum of S. Since initial anchor is good, we are able to bound comparisons used by OPT-MAX-LOW.

Observe that in OPT-MAX-LOW, we call SOFT-SEQ-ELIM three times in total: two times during NEAR-OPT-MAX and once to produce the final output. Each successive call of

SOFT-SEQ-ELIM acts on higher size, namely first we find $\varepsilon/4$-maximum in a small set and using this element as anchor, then we find $\varepsilon/2$-maximum in a larger set and finally using this new element as anchor, we find an ε-maximum of the whole set S.

Algorithm 22 OPT-MAX-LOW

inputs
  Set S, bias ε, confidence δ
Form a set Q by selecting $|S|^{3/4}$ random elements from S without replacement
$r \leftarrow$ NEAR-OPT-MAX$(Q, \frac{\varepsilon}{2}, \frac{\delta}{3})$
return SOFT-SEQ-ELIM$(S, r, \frac{\varepsilon}{2}, \varepsilon, \frac{\delta}{3})$

Lemma 55 bounds comparisons used by OPT-MAX-LOW and proves its correctness. Proof is in Appendix 4.C.6.

Lemma 55. *For* $\frac{1}{|S|^{1/3}} \geq \delta \geq \min(1/|S|, e^{-|S|^{1/4}})$, *w.p.*$\geq 1-\delta$, OPT-MAX-LOW$(S, \varepsilon, \delta)$ *uses* $O(\frac{|S|}{\varepsilon^2} \log \frac{1}{\delta})$ *comparisons and outputs r, an* ε*-maximum*

Higher ranges of confidence δ

For low ranges of confidence δ $\left(\delta \leq \frac{1}{|S|^{1/3}}\right)$, notice that $\log \frac{1}{\delta}$ and $\log \frac{|S|}{\delta}$ are of same order and hence if we use SOFT-SEQ-ELIM with a good anchor, we can guarantee complexity of $O\left(\frac{|S|}{\varepsilon^2} \log \frac{|S|}{\delta}\right) = O\left(\frac{|S|}{\varepsilon^2} \log \frac{1}{\delta}\right)$.

However, for high values of δ, this is not the case. We solve this problem by pruning S to a smaller set of size $|S|/\log|S|$ such that it contains all good elements and then use SOFT-SEQ-ELIM. Due to space constraint, we present PRUNE, the pruning algorithm, OPT-MAX-MEDIUM, and OPT-MAX-HIGH, linear complexity maxing algorithms for higher ranges of confidence in Appendix 4.C.8.

4.4.4 Full Algorithm

In Theorem 56 we bound comparisons used by OPT-MAX and prove its correctness. Proof follows from Lemmas 55 and corresponding Lemmas 67 and 68 for OPT-MAX-MEDIUM and OPT-MAX-HIGH given in Appendix 4.C.8.

Algorithm 23 OPT-MAX

inputs
Set S, bias ε, confidence δ
if $\delta \leq \frac{1}{|S|^{1/3}}$ **then**
return OPT-MAX-LOW(S, ε, δ)
end if
if $\delta \leq \frac{1}{\log |S|}$ **then**
return OPT-MAX-MEDIUM(S, ε, δ)
end if
return OPT-MAX-HIGH(S, ε, δ)

Theorem 56. *For* $\delta \geq \min(1/|S|, e^{-|S|^{1/4}})$, *w.p.*$\geq 1-\delta$, OPT-MAX$(S, \varepsilon, \delta)$ *uses* $O\left(\frac{|S|}{\varepsilon^2} \log \frac{1}{\delta}\right)$ *comparisons and outputs an* ε*-maximum of* S.

4.5 Ranking for SST+STI

[FOPS17] provides a ranking algorithm that w.p.$\geq 1 - 1/|S|$, uses $O\left(\frac{|S|}{\varepsilon^2} \log |S| (\log \log |S|)^3\right)$ comparisons and outputs an ε-ranking of input set S.

We build on their algorithm BINARY-SEARCH-RANKING, improving two components which lead to additional $(\log \log |S|)^3$ factor, thereby proposing an optimal ε-ranking algorithm that uses $O\left(\frac{|S|}{\varepsilon^2} \log |S|\right)$ comparisons.

In Appendix 4.5, we outline the algorithm proposed in [FOPS17], pointing out the two components that lead to additional factor, and present ideas that improve over these components. For detailed explanation of BINARY-SEARCH-RANKING we refer readers to [FOPS17]. Now we explain the high-level idea of how we improve over these components.

The two components that we improve upon share the property that each is being called for $\Omega(|S|/(\log |S|)^3)$ times and at each time finds a correct output w.p.$\geq 1 - 1/|S|^5$.

Instead of finding a correct output w.p.$\geq 1 - 1/|S|^5$ in one shot, and incurring high complexity, we propose the following. First use the component to find a correct output w.p.$\geq 1 - 1/\log |S|$, then check if the output is correct or not. If the output is deemed to be not correct,

run the component again, finding a correct output w.p.$\geq 1-1/|S|^6$.

Thus to show the potency of this idea, it suffices to show: One, the second run is only invoked a few times and two, the complexity of checking whether an output is correct is not high. Our main contribution is RANK-CHECK algorithm that checks if an ordered set is ε-ranked or not 3ε-ranked. We present RANK-CHECK in Appendix 4.D.3

Theorem 57. BINARY-SEARCH-RANKING(S,ε) *[FOPS17] with new improved components presented here, w.p.*$\geq 1-1/|S|$*, uses* $O\left(\frac{|S|\log|S|}{\varepsilon^2}\right)$ *comparisons and outputs an* ε*-ranking of S.*

4.6 Lower bound for ranking for MST+STI

In this section we show that there exists a model with both MST and STI properties under which any PAC ranking algorithm requires quadratic many comparisons. Consider the model $S=\{a_1,a_2,...,a_n\}$ s.t. a_1 is preferable to a_2 i.e., $\tilde{p}_{a_1,a_2}=1/2$ and comparison between any other pair is almost a fair coin flip i.e., $\tilde{p}_{a_i,a_j}=\mu\ \forall i<j$ and $\{i,j\}\neq\{1,2\}$ for some $\mu<1/n^{10}$. This model satisfies both MST and STI. Any permutation which has a_1 coming after a_2 is a $1/4$-ranking. But since comparison between any pair other than (a_1,a_2) is essentially a fair coin toss, any strategy that does not compare a_1 and a_2 will not have them in correct order in the output w.p.$\approx 1/2$ and hence won't be a $1/4$-ranking. Therefore this problem is similar to finding a single biased coin among $\binom{n}{2}$ coins which needs $\Omega(n^2)$ comparisons.

Theorem 58 bounds the complexity required for ε-ranking of models with MST and STI. Proof is in Appendix 4.E.

Theorem 58. *There exists a model with MST and STI properties for which any algorithm requires* $\Omega(n^2)$ *comparisons to output a* $1/4$*-ranking w.p.*$\geq 7/8$*.*

4.7 Finding pairwise probabilities for SST+STI

Theorem 57 shows that for a model satisfying both SST and STI, an ε-ranking can be found using $O\left(\frac{|S|\log|S|}{\varepsilon^2}\right)$ comparisons. In this section we answer the question whether under same model we can approximate all pairwise probabilities to accuracy of ε using almost same complexity.

We first show a lower bound of $\Omega\left(\frac{|S|\min(|S|,1/\varepsilon)}{\varepsilon^2}\log|S|\right)$ utilizing a model for which $\Omega(|S|\min(|S|,1/\varepsilon))$ pairwise probabilities need to be approximated using comparisons. Later we present APPROX-PROB that uses comparisons only for $O(|S|\min(|S|,1/\varepsilon))$ pairs and hence obtain orderwise same upper bound as lower bound.

4.7.1 Lower Bound

We show that any algorithm requires $\Omega\left(\frac{|S|\min(|S|,1/\varepsilon)\log|S|}{\varepsilon^2}\right)$ comparisons to approximate all pairwise probabilities to ε accuracy.

We prove the lower bound by using the model: $(4k+4)\varepsilon \le \tilde{p}_{a_{i+k},a_i} \le (4k+8)\varepsilon$ for $1 \le k \le \min(n-i, \lfloor\frac{1}{16\varepsilon}-2\rfloor)$ and $\tilde{p}_{a_{i+k},a_i} = 1/4$ for $k > \min(n-i, \lfloor\frac{1}{16\varepsilon}-2\rfloor)$.

It can be shown that this model satisfies both SST and STI. Under this model, the only way to approximate unfixed pairwise probabilities is by comparing those pairs. Since pairwise probabilities are not fixed for $\Omega(n\min(n,1/\varepsilon))$ pairs, any algorithm needs to approximate those many probabilities to accuracy of ε, hence the lower bound.

Theorem 59 bounds the required complexity to approximate all pairwise probabilities. Proof is in Appendix 4.F.1

Theorem 59. *For $\varepsilon < 1/48$, there exists a model that satisfies both SST and STI for which any algorithm requires $\Omega\left(\frac{|S|\min(|S|,1/\varepsilon)}{\varepsilon^2}\log|S|\right)$ comparisons to approximate all pairwise probabilities to ε accuracy w.p. $\ge 3/4$.*

4.7.2 Upper Bound

Here we propose an algorithm to approximate all pairwise probabilities to an accuracy of ε.

The proposed algorithm, first finds an $\varepsilon/8$-ranking of the input set S and then approximates pairwise probabilities. By Theorem 57, w.p.$\geq 1-\frac{1}{|S|^2}$ we can find an $\varepsilon/8$-ranking of the input set S using $O\left(\frac{|S|\log|S|}{\varepsilon^2}\right)$ comparisons. We present APPROX-PROB that given an $\varepsilon/8$-ranked set, approximates all pairwise probabilities to an accuracy of ε.

APPROX-PROB APPROX-PROB takes an $\varepsilon/8$-ranked ordered set S i.e., $\tilde{p}_{S(i),S(j)} \leq \varepsilon/8 \ \forall i < j$ and bias ε and approximates all pairwise probabilities to an accuracy of ε.

Note that it is enough to approximate $\tilde{p}_{S(j),S(i)}$ for $j \geq i$ since $\tilde{p}_{S(i),S(j)} = -\tilde{p}_{S(j),S(i)}$. For all $i > 1$, APPROX-PROB compares $S(i)$ and $S(1)$, $\frac{16\log|S|^4}{\varepsilon^2}$ times and approximates $\tilde{p}_{S(i),S(1)}$ by $\hat{\tilde{p}}_{S(i),S(1)}$, the fraction of times $S(i)$ won rounded off to the nearest multiple of ε. Since for perfectly ranked ordered set $\tilde{p}_{S(i+1),S(1)} \geq \tilde{p}_{S(i),S(1)}$, if $\hat{\tilde{p}}_{S(i+1),S(1)} < \hat{\tilde{p}}_{S(i),S(1)}$, then APPROX-PROB corrects $\hat{\tilde{p}}_{S(i+1),S(1)}$, setting it equal to $\hat{\tilde{p}}_{S(i),S(1)}$. It can be shown that $\tilde{p}_{S(i),S(1)}$ is approximated to an accuracy of $\frac{7\varepsilon}{8}$.

APPROX-PROB continues this process by approximating $\tilde{p}_{S(i),S(2)}$ for $i \geq 2$ by increasing i one at a time. For a perfectly ranked set, $\tilde{p}_{S(i-1),S(2)} \leq \tilde{p}_{S(i),S(2)} \leq \tilde{p}_{S(i),S(1)}$ and hence if $\hat{\tilde{p}}_{S(i-1),S(2)} = \tilde{p}_{S(i),S(1)}$, APPROX-PROB does not use comparisons to approximate $\tilde{p}_{S(i),S(2)}$, instead assigns $\hat{\tilde{p}}_{S(i),S(2)} = \hat{\tilde{p}}_{S(i-1),S(2)}$. Whenever $\hat{\tilde{p}}_{S(i-1),S(2)} \neq \tilde{p}_{S(i),S(1)}$, APPROX-PROB approximates $\tilde{p}_{S(i),S(2)}$ by comparing $S(i)$ and $S(2)$. It can be shown that $\tilde{p}_{S(i),S(2)}$ is approximated to accuracy of ε.

APPROX-PROB continues this process for $S(3)$, then $S(4)$ and so on until $S(n)$. Notice that whenever $\hat{\tilde{p}}_{S(i-1),S(j)} = \hat{\tilde{p}}_{S(i),S(j-1)}$, APPROX-PROB does not use comparisons to approximate $\tilde{p}_{S(i),S(j)}$ but simply assigns $\hat{\tilde{p}}_{S(i),S(j)} = \hat{\tilde{p}}_{S(i-1),S(j)}$. We show this in fact happens at many places and only $O(|S|\min(|S|,1/\varepsilon))$ pairwise probabilities are approximated using comparisons. This

enables obtaining orderwise same upper bound as the lower bound.

Algorithm 24 APPROX-PROB

inputs
  Ordered Set S, bias ε
$\hat{\tilde{p}}_{S(1),S(1)} = 0$
for i from 2 to $|S|$ **do**
  Compare $S(1)$ and $S(i)$ for $\frac{16}{\varepsilon^2}\log|S|^4$ times
  $\hat{\tilde{p}}_{S(i),S(1)} = \left[\frac{\text{fraction of times } S(i) \text{ won}}{\varepsilon} - \frac{1}{2}\right]\varepsilon$
  if $\hat{\tilde{p}}_{S(i),S(1)} < \hat{\tilde{p}}_{S(i-1),S(1)}$ **then**
    $\hat{\tilde{p}}_{S(i),S(1)} = \hat{\tilde{p}}_{S(i-1),S(1)}$
  end if
end for
for j from 2 to $|S|$ **do**
  $\hat{\tilde{p}}_{S(j),S(j)} = 0$
  for k from $j+1$ to $|S|$ **do**
    if $\hat{\tilde{p}}_{S(k-1),S(j)} = \hat{\tilde{p}}_{S(k),S(j-1)}$ **then**
      $\hat{\tilde{p}}_{S(k),S(j)} = \hat{\tilde{p}}_{S(k-1),S(j)}$
    else
      Compare $S(j)$ and $S(k)$ for $\frac{16}{\varepsilon^2}\log|S|^4$ times
      $\hat{\tilde{p}}_{S(k),S(j)} = \left[\frac{\text{fraction of times } S(k) \text{ won}}{\varepsilon} - \frac{1}{2}\right]\varepsilon$
    end if
  end for
end for

Theorem 60 shows the correctness of APPROX-PROB and bounds its comparisons. Proof is in Appendix 4.F.3

Theorem 60. *Given an* $\varepsilon/8$*-ranked ordered set S i.e.,* $\tilde{p}_{S(i),S(j)} \leq \varepsilon/8\ \forall i < j$*,* APPROX-PROB$(S,\varepsilon)$ *uses* $O(\frac{|S|\min(|S|,1/\varepsilon)}{\varepsilon^2}\log|S|)$ *comparisons and w.p.*$\geq 1 - \frac{1}{|S|^2}$ *approximates all pairwise probabilities to accuracy of* ε*.*

4.8 Experiments

In this section, we compare the performance of our maxing algorithms with previous work on synthetic data. All results presented here are averaged over 1000 runs.

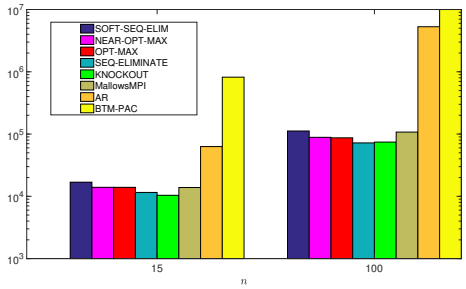


Figure 4.1: Maxing Algorithms for model with SST and STI

We compare our maxing algorithms SOFT-SEQ-ELIM, NEAR-OPT-MAX, and OPT-MAX with **SEQ-ELIMINATE** [FHO$^+$17], **KNOCKOUT** [FOPS17], **MallowsMPI** [BFHS14], **AR** [HSRW16] and **BTM-PAC** [YJ11]. **KNOCKOUT** and **BTM-PAC** are PAC maxing algorithms for models with both SST and STI properties. **SEQ-ELIMINATE** is a PAC maxing algorithm for SST model. **MallowsMPI**, originally designed for Mallows model, finds a condorcet winner which exists under WST. **AR** is a maxing algorithm that finds Borda winner that is same as condorcet winner under WST. In all experiments, we use maxing algorithms to find a 0.05-maximum with $\delta = 0.1$.

We first consider the model $p_{i,j} = 0.6\ \forall i < j$ same as in [YJ11, FOPS17, FHO$^+$17] that satisfies both SST and STI properties. Note that $i = 1$ is the only 0.05-maximum under this model. Figure 4.1 presents number of comparisons used by each maxing algorithm. Observe that compared to other algorithms, **BTM-PAC** uses too many comparisons even for $n = 15$. The reason might be **BTM-PAC** is mainly intended for reducing regret in the conventional bandits setting. The bar for **BTM-PAC** complexity for $n = 100$ is not fully shown in the figure to better scale the other complexity bars. Comparison complexity of AR is high for $n = 100$ mainly because **AR** eliminates elements based on Borda scores and Borda scores are very close to each other for large n. We drop **BTM-PAC** and **AR** henceforth.

Now we consider a model that satisfies MST but not SST, i.e., $p_{5i+l,5i+k} = 0.6\ \forall i < n/5-1$, $1 \leq l < k \leq 5$ and $p_{5i+l,5j+k} = 0.52\ \forall i < j < n/5-1$, $0 < l,k \leq 5$. Notice that under this model elements are divided into groups of five where within each group $|\tilde{p}_{i,j}| = 0.1$ and for elements

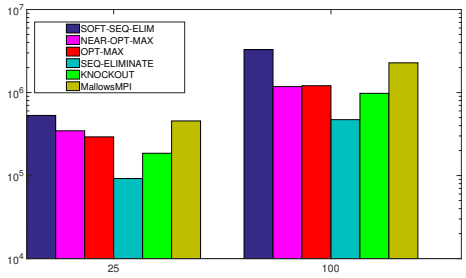


Figure 4.2: Maxing Algorithms for model with MST but not SST

in two different groups $|\tilde{p}_{i,j}| = 0.02$, hence there is a 0.05-maximum in each group. Figure 4.2 demonstrates comparison complexity of algorithms under this model. **SEQ-ELIMINATE** uses fewer comparisons, but it fails to output a 0.05-maximum with probability 0.21 for $n = 25$ and 0.19 for $n = 100$. Hence **SEQ-ELIMINATE** fails once SST is not satisfied. This is because when you compare a 0.05-maximum of a group with an element in other group, 0.05-maximum can get eliminated with probability ≈ 0.5. Hence with lots of groups **SEQ-ELIMINATE** fails. Other algorithms find a 0.05-maximum in all runs. We drop **SEQ-ELIMINATE** henceforth.

Now we consider a model that does not satisfy STI but satisfies MST i.e., $n = 10$ and $p_{1,j} = 1/2 + \tilde{q}\ \forall j \leq n/2$, $p_{1,j} = 1\ \forall j > n/2$ and $p_{i,j} = 1/2 + \tilde{q}\ \forall 1 < i < j$, $\tilde{q} < 0.05$. Under this model any $i \leq 5$ is a 0.05-maximum. Figure 4.3 shows the average comparison complexity of algorithms under this model. **KNOCKOUT** uses fewer comparisons, but fails to output a 0.05-maximum with probability 0.12 for $\tilde{q} = 0.001$ and 0.25 for $\tilde{q} = 0.0001$, hence fails to meet the confidence requirement once STI is dropped. Other algorithms find a 0.05-maximum in all runs.

It is interesting to note that **MallowsMPI** uses more comparisons as $\tilde{q}$ decreases, whereas the complexity of other algorithms remains almost same. This is because **MallowsMPI** tries to find absolute maximum which is not always practical. Further note that the performance of SOFT-SEQ-ELIM is better than NEAR-OPT-MAX, and NEAR-OPT-MAX is better than OPT-MAX. This is because the bias gap for SOFT-SEQ-ELIM, NEAR-OPT-MAX and OPT-MAX is ε, $\varepsilon/2$ and $\varepsilon/4$ respectively, resulting in higher constants for NEAR-OPT-MAX and OPT-MAX. While the

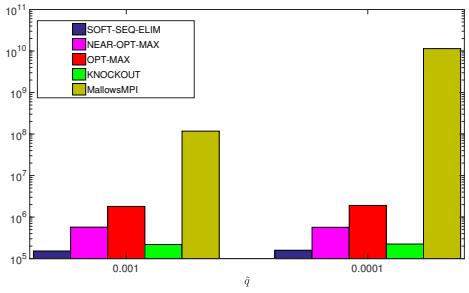


Figure 4.3: Maxing algorithms for model without STI

theoretical order complexity is higher for SOFT-SEQ-ELIM, in practice it can find a good anchor quickly and seems to have near-linear order complexity.

4.9 Conclusion

We studied the problem of maxing, ranking, and estimating comparison probabilities under different stochastic transitivity constraints. We showed that under WST, maxing needs quadratic comparisons. We also presented a linear-complexity algorithm for maxing under MST. We also proposed an optimal ranking algorithm for SST models with Stochastic Triangle Inequality, closing $(\log\log n)^3$ gap. For the same model, we proposed an optimal algorithm for estimating the comparison probabilities.

4.10 Acknowledgement

Chapter 4, in full, is a reprint of the material as it appears in *International Conference on Machine Learning*. Falhatagar, Moein, Ayush, Jain, Alon Orlitsky, Venkatadheeraj Pichapati, and Vaishakh Ravindrakumar, 2018. The dissertation author was the primary investigators and author of this paper.

4.A Lower bound for WST

Outline

We first present a model that satisfies WST and relate it to a *linear jigsaw puzzle* such that lower bound on jigsaw puzzle implies lower bound for finding an $1/4$-maximum under the model.

Consider the following model with n elements $S = \{1, 2, \ldots, n\}$: $\tilde{p}_{i,i+1} = \frac{1}{2} \forall i < n$, and $\tilde{p}_{i,j} = \mu (0 < \mu < 1/n^{10}), \forall j > i+1$. This model satisfies WST since there exists an underlying order $\succ$, $i \succ j$ if $i < j$ (because $\tilde{p}_{i,j} > 0$). Observe that 1 is the only $1/4$-maximum under this model.

We prove the Lemma by reducing the above model to the model where μ is replaced by 0.

Note that μ is so small that if we consider a model where we replace μ with 0, the comparisons behave essentially similarly. More formally, let M_μ be the model considered above and M_0 be the model when μ is replaced with 0. Let C denote a sequence of comparisons where each element of the sequence includes the elements compared and its outcome. Further, for each sequence C, let $P_\mu(C)$ and $P_0(C)$ denote the probability of sequence C under models M_μ and M_0 respectively. Now consider a sequence C of comparisons of length $\leq n^2/20$. Then

$$\frac{P_0(C)}{P_\mu(C)} \geq \left(\frac{1/2}{1/2+\mu}\right)^{n^2/20} \geq e^{-n^2/(10n^{10})} \geq \frac{6}{7}$$

Thus the probability of any sequence of length $\leq \frac{n^2}{20}$ is approximately same under both models. Hence if there is an algorithm that uses $\leq \frac{n^2}{20}$ comparisons and w.p.$\geq 7/8$ produces the $1/4$-maximum under M_μ model then applying same algorithm over M_0 model produces the $1/4$-maximum w.p.$\geq \frac{7}{8} \cdot \frac{6}{7} = \frac{3}{4}$. Hence, lower bound of $\Omega(n^2)$ over M_0 model implies a lower bound of $\Omega(n^2)$ over M_μ model.

We now show that under M_0 model, any algorithm requires $\Omega(n^2)$ comparisons to find the $1/4$-maximum w.p.$\geq \frac{3}{4}$, thus proving the Lemma. From now, we only consider model M_0.

Notice that under M_0, whenever two non adjacent elements are compared i.e., i and j with $|i-j|>1$, the comparison output is a fair coin toss. We make the problem simpler by revealing the extra information of whether elements that are being compared are adjacent or not. Notice that this only makes the problem easier, namely, complexity for modified problem is smaller than that of original problem.

The modified problem is similar to a linear jigsaw puzzle where if we ask a question about two pieces we will know if pieces are adjacent or not and if adjacent, which piece is on the left and the goal is to find the left most piece. To make the proof simpler, we change the question model slightly. In the new model, when we question about ordered pair (e,f), we get to know if e is left neighbor of f. Notice that we can simulate one question in previous model by asking two questions (e,f) and (f,e) in new model.

Now we present proof to show that for linear jigsaw puzzle, any algorithm requires $\Omega(n^2)$ comparisons to find the left most piece with probability $\geq 3/4$. This implies the Lemma.

4.A.1 Lower bound for Jigsaw Puzzle

Outline: We first describe the main idea briefly. Notice that we start with n unconnected components where each piece refers to an unconnected component. Each new connection (two pieces are connected if they are neighbours) revealed between the pieces reduces the number of unconnected components by 1. We first show that for some small constant c, even after asking $c^2n^2/2$ questions, w.h.p., $< 20cn$ connections are revealed. For this, we divide pieces into two groups based on number of questions asked about them. Group 1: pieces for which $\leq cn$ questions are asked and Group 2: pieces for which $> cn$ questions are asked. Then we bound the probability that a question between two pieces from Group 1 results in finding a new connection by $8/n$. Therefore, by asking $c^2n^2/2$ questions w.h.p., we will find $\leq 16cn$ such connections. Since total number of questions asked is $c^2n^2/2$, number of pieces about which we have asked $> cn$ questions is $< cn$. Hence w.h.p. total number of connections found using $c^2n^2/2$ comparisons is

$< 16cn + 2 \cdot cn < 20cn$.

Since total connections found is $< 20cn$ there are $> (1 - 20c)n$ unconnected components. We show that all unconnected components' leftmost pieces for which we asked less than $\leq cn$ questions are all almost equally likely to be the leftmost piece of the entire puzzle. Hence returning such a piece as the leftmost piece of puzzle will give accuracy of only $O(1/n)$. Further we show that the probability that out of $c^2n^2/2$ total questions, we have asked $> cn$ questions about the left most piece, is bounded by $2c$. This results in an upper bound of $3/4$ on probability of success in finding the left most piece.

Setup: We have n elements from 1 to n. Let $\mathcal{P}$ denotes the set of all possible permutations of elements in $[n]$. Note $|\mathcal{P}| = n!$. Adversary chooses a permutation from $\mathcal{P}$ randomly uniformly. And we want to know the top element in the permutation chosen by adversary.

Let $q_{(i,j)}$ corresponds to the question whether i is left neighbour of j and ξ be the set of all questions i.e., $\xi = \{q_{(i,j)} : i \neq j \text{ and } i, j \in [n]\}$.

Let $A(q_{(i,j)})$ denotes the answer of question $q_{(i,j)}$ which takes the value 1, if the answer is "yes" (i is left neighbor of j) and 0 if it is "no". Note that $|\xi| = n(n-1)$, and for any chosen permutation, exactly $n-1$ of these have yes answers.

At each time step we ask a question $q_t = q_{(l_t,r_t)} \in \xi$ and get answer $A_t = A(q_t)$. An algorithm is an strategy to decide which question is to be asked next based on the response of questions asked in past.

Formally any algorithm $\mathbb{A}$ consists of a sequence of functions f_t, such that $q_t = f_t(q_1^{t-1} \times A_1^{t-1}) \in \xi$, we allow f_ts to be random functions. For any reasonable algorithm we can assume $q_t \neq q_k$ for $k \in [t-1]$, i.e. it will not ask same question twice.

We denote set of all the questions asked till time t by $Q_t \triangleq \{q_k \; \forall \; k \in [t]\}$. Let $Y(t) = \{q_j : A(q_j) = 1, \; j \in [t]\}$, denotes the collection of questions asked till time t, which resulted in "yes" answers (or new connections).

Let $\mathcal{P}_t$ be the set of all valid permutations at time t. At start, $\mathcal{P}_0 = \mathcal{P}$. When we ask $q_t = q_{(l_t,r_t)}$, if $A_t = 1$, then to get $\mathcal{P}_t$ from $\mathcal{P}_{t-1}$, we remove all such permutations from $\mathcal{P}_{t-1}$ in which l_t is not the left neighbour of r_t, and if $A_t = 0$ we do the opposite i.e. remove all such permutations from $\mathcal{P}_{t-1}$ in which l_t is left neighbour of r_t to get $\mathcal{P}_t$. After asking t questions all permutations in $\mathcal{P}_t$ have equal chances of being the correct permutation (Since posterior distribution will have equal probability for all the valid permutations, because prior distribution was uniform).

We divide the elements into two groups based on number of questions asked about the element. For that, let $n_i(t) \triangleq |\{k \in [t] : i = l_k \text{ or } i = r_k\}|$, denotes the number of questions asked, which involves element i.

Let $T = c^2n^2/2$, with $c = 1/1000$. From now on we assume $t \leq T$. Let $\chi(t) \triangleq \{i : |n_i(t)| > cn\}$, which is collection of all the elements about which we asked more than cn questions till time t. Then from pigeonhole principle we have:

$$|\chi(t)| \leq cn.$$

Define indicator random variable $I(t)$ which takes values 1, if $A(q(t)) = 1$ and $l_t \notin \chi(t-1)$ and $r_t \notin \chi(t-1)$. Then, we can bound the total number of yes answers $|Y(t)|$ by the sum of $I(k)$ upto time t and size of $\chi(t)$.

$$|Y(t)| \leq \sum_{k\in[t]} I(k) + 2|\chi(t)| \leq \sum_{k\in[t]} I(k) + 2cn,$$

where first inequality follows from: each element in $\chi(t)$ can increase the count of yes answers by 2 at most.

Now we bound the total number of yes answers by $20cn$.

Lemma 61. *For any algorithm* $\mathbb{A}$, $T = c^2n^2/2$, $\Pr[|Y(T)| > 20cn] < 1/4$.

Proof. To show that $|Y(T)| \leq 20cn$, it is enough to show that $\sum_{k\in[T]} I(k) \leq 18cn$.

Let us count the time steps t for which $I(t) = 1$ as success. Then we show that probability of success in a time step is upper bounded by $8/n$ until first $18cn$ successes. Then number of steps taken for k^{th} success for $k < 18cn$ can be thought as geometric distribution with mean $> n/8$. Therefore to get $18cn$ successes we need more than $\sim 18cn \times n/8 > 2cn^2$ steps, and probability of getting as many successes is $c^2n^2/2$, with c small enough is $< 1/4$ (follows from properties of negative binomial distribution, which is sum of geometric distributions).

We complete the proof by showing that

$$\Pr[I(t) = 1 \mid \sum_{k\in[t-1]} I(k) < 18cn] < 8/n \tag{4.1}$$

Lets assume $\sum_{k\in[t-1]} I(k) < 18cn$.

Then we prove that for any pair $\alpha, \beta \in [n]$ such that $\alpha, \beta \notin \chi(t-1)$ and $q_{(\alpha,\beta)} \notin Q_{t-1}$, the fraction of permutations in $\mathcal{P}_{t-1}$, in which α is left neighbour of β is $\leq 8/n$.

Let B_1 be the collection of all the permutations in $\mathcal{P}_{t-1}$ in which α is left neighbour of β and α is ranked in top $n/2$ elements. Similarly, let B_2 be the collection of all the permutations in $\mathcal{P}_{t-1}$ in which α is left neighbour of β and α is ranked in bottom $n/2$ elements. We first show that $|\mathcal{P}_{t-1}| \geq |B_1|n/4$.

For each permutation in B_1, we show a set of permutations of size $> n/4$ such that all sets are disjoint and no permutation has α as left neighbour of β. We consider following permutation, which is a member of B_1.

$$b_y, b_{y-1}, ...b_2, b_1, \alpha, \beta, a_1, a_2, a_3, ..., a_{x-1}, a_x$$

Here $x + y + 2 = n$ and $x > n/2 - 2$.

Let u be the minimum index such that algorithm has asked less than cn questions about a_{u-1} and questions (α, a_u), (a_{u-1}, a_u) are not asked before time t i.e.,

$u = \min\{i > 1 : a_{i-1} \notin \chi(t-1)$ and $q_{(\alpha,a_i)} \notin Q_{t-1}$ & $q_{(a_{i-1},a_i)} \notin Q_{t-1}\}$.

Note that $u \leq 1 + |\{i : a_{i-1} \in \chi(t-1)\}| +$

$|\{i : q_{(\alpha,a_i)} \in Q_{t-1}\}| + |\{i : q_{(a_{i-1},a_i)} \in Q_{t-1}\}|$.

Observe that:

$$|\{i : a_{i-1} \in \chi(t-1)\}| \leq |\chi(t-1)| \leq cn \tag{4.2}$$

and

$$|\{i : q_{(\alpha,a_i)} \in Q_{t-1}\}| \leq n_\alpha(t-1) \leq cn. \tag{4.3}$$

If a_{i-1} is right neighbour of a_i in a valid permutation in $\mathcal{P}_t - 1$, then if $q_{(a_{i-1},a_i)}$ has been asked then it would have resulted in answer "yes" because if answer was "no" then they can't be neighbours in a valid permutation. Hence $|\{i : q_{(a_{i-1},a_i)} \in Q_{t-1}\}|$ is upper bounded by $|Y(t-1)|$. Combining these and with our assumption above that $\sum_{k\in[t-1]} I(k) \leq 18cn$, we get,

$$u \leq 1 + cn + cn + 2cn + \sum_{k\in[t-1]} I(k) \leq 1 + 22cn \leq n/32. \tag{4.4}$$

Let Λ be the set of indices $i > u+1$ such that questions $q_{(\alpha,a_i)}$, $q_{(a_{u-1},a_i)}$, $q_{(a_{i-1},a_i)}$ and $q_{(a_{i-1},\beta)}$ are not asked before time $t-1$ i.e.,

$\Lambda = \{i > u+1 : q_{(\alpha,a_i)} \notin Q_{t-1}$ & $q_{(a_{(u-1)},a_i)} \notin Q_{t-1}$ & $q_{(a_{i-1},a_i)} \notin Q_{t-1}$ & $q_{(a_{i-1},\beta)} \notin Q_{t-1}\}$.

Notice that

$$\begin{aligned}|\Lambda| \geq\ & x - u - 1 - |\{i : q_{(\alpha,a_i)} \in Q_{t-1}\}| \\ & - |\{i : q_{(a_{(u-1)},a_i)} \in Q_{t-1}\}| - |\{i : q_{(a_{i-1},a_i)} \in Q_{t-1}\}| \\ & - |\{i : q_{(a_{i-1},\beta)} \in Q_{t-1}\}|.\end{aligned}$$

Observing $x > n/2-2$, then following the steps similar to the proof of equation (4.4) we get:

$$\begin{aligned}
|\Lambda| \geq\ & n/2-2-n/32-1-n_{\alpha}(t-1) \\
& -n_{a_{(u-1)}}(t-1)-|Y(t-1)|-n_{\beta}(t-1) \\
\geq & n/2-2-n/32-1-cn-cn-20cn-cn \\
\geq & 15n/32-3-23cn \\
\geq & n/4.
\end{aligned}$$

For any $v \in \Lambda$, we claim that following permutation will also be part of $\mathcal{P}_{t-1}$:

$$\begin{aligned}
&\{b_y, b_{y-1}, ..., b_2, b_1, \alpha\}, \{a_u, a_{u+1}, ..., a_{v-1}\}, \\
&\{\beta, a_1, a_2, ..., a_{u-1}\}, \{a_v, a_{v+1}, ...a_x\}.
\end{aligned}$$

In above permutation we have put curly parentheses to highlight the changes made from original permutation. To show that the permutation above is valid, i.e., lies in $\mathcal{P}_{t-1}$, we need to show that (α, a_u), (a_{v-1}, β) and (a_{u-1}, a_v) are valid connections,which is easy to do from the definition of u and Λ. We skip the details here.

The original permutation can be uniquely recovered by finding the link (a_{u-1}, a_v), which can be found using the fact that (a_{u-1}, a_v) is the first set of consecutive elements (i, j) which are ranked after β such that $q_{(i,j)} \notin Q_{t-1}$, $q_{(\alpha,j)} \notin Q_{t-1}$ and $i \notin \chi(t-1)$ (this again can be verified from the definition of u and Λ). Therefore, to each permutation in B_1, we can map a disjoint set of $n/4$ permutations, which are part of $\mathcal{P}_{t-1}$ and in which α is not the right neighbour of β. Hence $|\mathcal{P}_{t-1}| \geq |\Lambda| \times |B_1| \geq |B_1| n/4$. Similarly, it can be shown that $|\mathcal{P}_{t-1}| \geq |B_2| n/4$. Therefore,

$$\Pr[A(q_{(\alpha,\beta)}) = 1 | \sum_{k\in[t-1]} I(k) < 18cn]$$
$$= \frac{|B_1| + |B_2|}{|\mathcal{P}_{t-1}|} \leq 8/n.$$

Thus we get:

$$\Pr[I(t) = 1 | \sum_{k\in[t-1]} I(k) < 18cn] \leq 8/n.$$

□

Next, we prove that if $\beta \notin \chi(t-1)$ and $|Y(t-1)| \leq 20cn$, then Pr[β is top element in permutation] $\leq 2/n \; \forall \, t \leq c^2n^2/2$. We use the same idea as earlier in the proof, to each permutation in $\mathcal{P}_{t-1}$ with β as top element, we can map a disjoint set of $n/2$ permutations which are also part of $\mathcal{P}_{t-1}$, and don't have β at the top.

Consider the following permutation of $\mathcal{P}_{t-1}$, in which β is a top element.

$$\beta, a_1, a_2, a_3, ..., a_{n-2}, a_{n-1}$$

Let $\widehat{u}$ be the minimum index such that question $q_{(a_{\widehat{u}-1}, a_{\widehat{u}})}$ is not asked before and algorithm didn't ask more than cn questions about $a_{\widehat{u}-1}$ i.e.,

$\widehat{u} = \min\{i > 1 : a_{i-1} \notin \chi(t-1)$ and $q_{(a_{i-1},a_i)} \notin Q_{t-1}\}$.

Note that

$\widehat{u} \leq 1 + |\{i : a_{i-1} \in \chi(t-1)\}| + |\{i : q_{(a_{i-1},a_i)} \in Q_{t-1}\}|$.

As before

$$\widehat{u} \leq 1 + cn + |Y(t-1)| < 1 + cn + 20cn < n/16.$$

Let $\widehat{\Lambda}$ be the set of all indices $i > \widehat{u}+1$ such that questions $q_{(a_{(\widehat{u}-1)},a_i)}$, $q_{(a_{i-1},a_i)}$, $q_{(a_{i-1},\beta)}$ are not asked before i.e., $\widehat{\Lambda} = \{i > \widehat{u}+1 : q_{(a_{(\widehat{u}-1)},a_i)} \notin Q_{t-1} \ \& \ q_{(a_{i-1},a_i)} \notin Q_{t-1} \ \& \ q_{(a_{i-1},\beta)} \notin Q_{t-1}\}$. Then observe that

$$\begin{aligned}
|\widehat{\Lambda}| &\geq n-2-\widehat{u}-|\{i : q_{(a_{u-1},a_i)} \in Q_{t-1}\}| \\
&\quad -|\{i : q_{(a_{i-1},a_i)} \in Q_{t-1}\}|-|\{i : q_{(a_{i-1},\beta)} \in Q_{t-1}\}| \\
&\geq n-2-n/16-cn-|Y(t-1)|-cn > n/2
\end{aligned}$$

As before, for any $\widehat{v} \in \widehat{\Lambda}$, we claim that following permutation will also be part of $\mathcal{P}_{t-1}$:

$$\begin{aligned}
&\{a_{\widehat{u}}, a_{\widehat{u}+1}, ..., a_{\widehat{v}-1}\}, \{\beta, a_1, a_2, ..., a_{\widehat{u}-2}, a_{\widehat{u}-1}\}, \\
&\{a_{\widehat{v}}, a_{\widehat{v}+1}, ...a_{n-1}\}
\end{aligned}$$

And it is easy to verify that from this we can get the original permutation back uniquely. Hence, we have showed that if $|Y(t-1)| \leq 20cn$, then for any $\beta \notin \chi(t-1)$, less than $2/n$ fraction of all the permutation $\mathcal{P}_{t-1}$ contains β as the top element. Therefore, at the end if algorithm returns an element $\beta \notin \chi(T)$ as top element, probability of it being correct is upper bounded by $2/n$. And if we predict the top element β such that $\beta \in \chi(T)$, then success probability is upper bounded by the probability that $\chi(T)$ contains the top element. At $t = 0$, $\chi(0)$ is empty. New elements are added in $\chi(.)$ as we ask new questions. When we asked $cn+1$'th question about that element, from the previous discussion at that point probability of that element being top element is upper bounded by $2/n$. And we have at most cn elements in $\chi(T)$, therefore:

$$\begin{aligned}
&\Pr\left[\chi(T) \text{ contains top element} \middle| |Y(t-1)| \leq 20cn\right] \\
&\leq cn \times 2/n = 2c.
\end{aligned}$$

Therefore, probability that our prediction of top element is correct:

$$
\begin{aligned}
&\Pr[\text{Algorithm returns correct top element}] \\
\leq &\Pr[|Y(T)| > 20cn] \\
&+ \Pr[\chi(T) \text{ contains top element} | |Y(t-1) \leq 20cn] \\
&+ \Pr[\text{Algorithm returns } \beta \notin \chi(T) \\
&\text{ and is correct} | Y(t-1)| \leq 20cn] \\
&\leq 1/4 + 2c + 2/n < 3/4.
\end{aligned}
$$

4.B Estimating pairwise probabilities for WST

4.B.1 BRUTE-FORCE

For WST model, [FHO+17] presented a PAC-ranking algorithm that uses $O\left(\frac{n^2}{\varepsilon^2}\log\frac{n}{\delta}\right)$ comparisons. In the process, they present a trivial algorithm to estimate all pairwise probabilities to accuracy of ε using $O\left(\frac{n^2\log n}{\varepsilon^2}\right)$ comparisons. We present their algorithm here for completeness.

EST-PROB

EST-PROB(i,j,ε,δ) compares i and j for $\frac{1}{2\varepsilon^2}\log\frac{2}{\delta}$ times and returns the fraction of times i won. With probability $\geq 1-\delta$, this fraction approximates $p_{i,j}$ to an accuracy of ε.

Algorithm 25 EST-PROB

1: **inputs**
2: element i, element j, bias ε, confidence δ.
3: Compare i and j for $\frac{1}{2\varepsilon^2}\log\frac{2}{\delta}$ times.
4: **return** Fraction of times i won

Lemma 62. EST-PROB(i,j,ε,δ) *uses* $\frac{1}{2\varepsilon^2}\log\frac{2}{\delta}$ *comparisons and w.p.*$\geq 1-\delta$ *approximates* $p_{i,j}$ *to an additive error of* ε.

Proof. Proof follows from Hoeffding's inequality. □

BRUTE-FORCE

BRUTE-FORCE(S,ε,δ) approximates all pairwise probabilities $p_{i,j}$ using EST-PROB$(i,j,\varepsilon,\frac{2\delta}{|S|(|S|-1)})$. Observe that w.p.$\geq 1-\frac{2\delta}{|S|(|S|-1)}$, EST-PROB$(i,j,\varepsilon,\frac{2\delta}{|S|(|S|-1)})$ approximates $\tilde{p}_{i,j}$ to an accuracy of ε. Hence by union bound, w.p.$\geq 1-\delta$, BRUTE-FORCE(S,ε,δ) approximates all pair-wise probabilities to an accuracy of ε.

Algorithm 26 BRUTE-FORCE

1: **inputs**
2: Set S, bias ε, confidence δ
3: **for** every pair $\{i,j\}$ such that $i,j \in S$ **do**
4: $\hat{p}(i,j) \leftarrow$ EST-PROB$(i,j,\frac{\varepsilon}{2},\frac{2\delta}{n(n-1)})$
5: $\hat{p}(j,i) \leftarrow 1-p(i,j)$
6: **end for**

In the below Lemma, we bound complexity of BRUTE-FORCE and prove its correctness.

Lemma 63. BRUTE-FORCE(S,ε,δ) *uses* $O(\frac{n^2}{\varepsilon^2}\log\frac{n}{\delta})$ *comparisons and w.p.*$\geq 1-\delta$ *approximates all pairwise probabilities to accuracy of* ε.

Notice that under WST, once we all pairwise probabilities are approximated to accuracy of $\varepsilon/2$, one can find ε-maximum and ε-ranking.

4.C PAC maxing for MST

4.C.1 Property of MST

We first prove a property of MST that helps us in bounding comparisons for MST.

Lemma 64. *Under MST, if* $\varepsilon > 0$, $\tilde{p}_{i,j} \leq \varepsilon$, $\tilde{p}_{k,j} > \varepsilon$ *then* $\tilde{p}_{i,k} \leq \varepsilon$.

Proof. We assume that $\tilde{p}_{i,k} > \varepsilon$ and prove Lemma by contradiction.

Since $\tilde{p}_{i,k} > \varepsilon$ and $\tilde{p}_{k,j} > \varepsilon$, then $i \succ k \succ j$ and hence by MST,

$$\tilde{p}_{i,j} \geq \min(\tilde{p}_{i,k}, \tilde{p}_{k,j}) > \varepsilon$$

which contradicts the Lemma statement.

Hence $\tilde{p}_{i,k} \leq \varepsilon$. □

4.C.2 COMPARE

For better performance in practice COMPARE stops earlier if $\tilde{p}_{i,j} \ll \varepsilon_l$ or $\tilde{p}_{i,j} \gg \varepsilon_u$. This step does not affect our bounds.

Algorithm 27 COMPARE

inputs

element i, element j, lower bias $\varepsilon_l \geq 0$, upper bias $\varepsilon_u > \varepsilon_l$, confidence δ

initialize

$\varepsilon_m = \frac{\varepsilon_l + \varepsilon_u}{2}$, $\hat{\tilde{p}}_{i,j} \leftarrow 0$, $\hat{c} \leftarrow \frac{1}{2}$, $t \leftarrow 0$, $w \leftarrow 0$

while $|\hat{\tilde{p}}_{i,j} - \varepsilon_m| \leq \hat{c} + (\varepsilon_u - \varepsilon_l)/4$ and $t \leq \frac{8}{(\varepsilon_u - \varepsilon_l)^2} \log \frac{2}{\delta}$ **do**

Compare i and j

if i wins **then**

$w \leftarrow w + 1$

end if

$t \leftarrow t + 1$

$\hat{\tilde{p}}_{i,j} \leftarrow \frac{w}{t} - \frac{1}{2}$, $\hat{c} \leftarrow \sqrt{\frac{1}{2t} \log \frac{4t^2}{\delta}}$

end while

if $\hat{\tilde{p}}_{i,j} < (\varepsilon_l + \varepsilon_m)/2$ **then**

return 1

end if

if $\hat{\tilde{p}}_{i,j} > (\varepsilon_m + \varepsilon_u)/2$ **then**

return 3

end if

return 2

Proof for Lemma 50

Proof. We first bound the number of comparisons.

Notice that COMPARE$(i, j, \varepsilon_l, \varepsilon_u, \delta)$ compares elements i and j for at most $m = \frac{8}{(\varepsilon_u - \varepsilon_l)^2} \log \frac{2}{\delta}$ times and hence bound on comparisons follows.

We first show correctness for case of $\tilde{p}_{i,j} \leq \varepsilon_l$.

Let $\hat{p}^t_{i,j}$ and $\hat{c}^t$ denote $\hat{p}_{i,j}$ and $\hat{c}$ respectively after t comparisons between i and j during COMPARE$(i,j,\varepsilon_l,\varepsilon_u,\delta)$. COMPARE$(i,j,\varepsilon_l,\varepsilon_u,\delta)$ does not output 1 only if $\hat{p}^t_{i,j} \geq \frac{1}{2} + \frac{\varepsilon_l+3\varepsilon_u}{4} + \hat{c}^t$ for any $t < m = \frac{8}{(\varepsilon_l-\varepsilon_u)^2}\log\frac{2}{\delta}$ or if $\hat{p}^m_{i,j} > \frac{1}{2} + \frac{3\varepsilon_l+\varepsilon_u}{4}$. We bound the probability of either of these events by $\frac{\delta}{2}$ and the result follows from the union bound.

By Hoeffding's inequality,

$$\begin{aligned} Pr\left(\hat{p}^t_{i,j} > \frac{1}{2} + \frac{\varepsilon_l+3\varepsilon_u}{4} + \hat{c}^t\right) &\leq Pr\left(\hat{p}^t_{i,j} > \frac{1}{2} + \varepsilon_l + \hat{c}^t\right) \\ &\leq e^{-2t(\hat{c}^t)^2} \\ &= e^{-\log\frac{4t^2}{\delta}} \\ &= \frac{\delta}{4t^2}. \end{aligned}$$

By the union bound, $Pr\left(\exists t \text{ s.t. } \hat{p}^t_{i,j} > \frac{1}{2} + \frac{\varepsilon_l+3\varepsilon_u}{4} + \hat{c}^t\right) \leq \sum_t \frac{\delta}{4t^2} \leq \frac{\delta}{2}$.

Similarly, by Hoeffding's inequality,

$$\begin{aligned} Pr\left(\hat{p}^m_{i,j} > \frac{1}{2} + \frac{3\varepsilon_l+\varepsilon_u}{4}\right) &\leq e^{-2m((\varepsilon_u-\varepsilon_l)/4)^2} \\ &= e^{-\log\frac{2}{\delta}} \\ &= \frac{\delta}{2}. \end{aligned}$$

Hence if $\tilde{p}_{i,j} < \varepsilon_l$, w.p.$\geq 1-\delta$, COMPARE outputs 1. We can prove similarly for cases $\tilde{p}_{i,j} > \varepsilon_u$, $\varepsilon_l \leq \tilde{p}_{i,j} \leq (\varepsilon_l+\varepsilon_u)/2$ and $(\varepsilon_u+\varepsilon_l)/2 < \tilde{p}_{i,j} \leq \varepsilon_u$. □

4.C.3 Proof for Lemma 51

Proof. Observe that COMPARE is called for at most $|S|(|S|-1)/2$ times. Since when calling COMPARE, we use confidence parameter of $\frac{2\delta}{|S|^2}$, the probability that COMPARE gives expected answer always is $\geq 1-\delta$. From now, we assume that COMPARE$(i,j,\varepsilon_l,\varepsilon_u,2\delta/|S|^2)$ always

returns 1 if $\tilde{p}_{i,j} \leq \varepsilon_l$, 1 or 2 if $\varepsilon_l < \tilde{p}_{i,j} \leq (\varepsilon_l + \varepsilon_u)/2$, 2 or 3 if $(\varepsilon_l + \varepsilon_u)/2 < \tilde{p}_{i,j} \leq \varepsilon_u$ and 3 if $\tilde{p}_{i,j} > \varepsilon_u$.

Let r^1 be the value of anchor r in the beginning and r^t be the value of r after $t-1$ changes of r.

We first show that $\tilde{p}_{r^t,r^l} > (\varepsilon_l + \varepsilon_u)/2$ for all $l < t$. Since $r^{t+1} = e$ only if $\textsc{Compare}(e, r^t, \varepsilon_l, \varepsilon_u, \frac{2\delta}{|S|^2})$ returns 3, $\tilde{p}_{r^{t+1},r^t} > (\varepsilon_l + \varepsilon_u)/2$. Hence by MST, $\tilde{p}_{r^t,r^l} > (\varepsilon_l + \varepsilon_u)/2$ for all $l < t$.

We now bound the number of comparisons. We first bound the number of Compare calls during Soft-Seq-Elim. Let T be the set of elements for which r^1 is ε_l-preferable i.e., $T \stackrel{\text{def}}{=} \{e : \tilde{p}_{e,r^1} \leq \varepsilon_l\}$. Since r^1 is an (ε_l, m)-*good anchor* element, $|T| \geq |S| - m$. Notice that for $e \in T$, $\tilde{p}_{e,r^1} \leq \varepsilon_l$ and since $\tilde{p}_{r^t,r^1} > (\varepsilon_u + \varepsilon_l)/2 > \varepsilon_l$ by MST and Lemma 64, $\tilde{p}_{e,r^t} \leq \varepsilon_l \ \forall t$. Hence $\textsc{Compare}(e, r^t, \varepsilon_l, \varepsilon_u, 2\delta/|S|^2)$ returns 1 for all t and therefore, number of Compare calls spent on e is 1. Thus number of Compare calls spent on set T is $|T|$. Since elements in T will not become anchors, there are atmost $|S| - |T| \leq m$ rounds and hence number of Compare calls spent on an element in $S \setminus T$ is $\leq m$. Therefore total number of Compare calls is

$$\leq |T| + m|S \setminus T| \leq |S| + m^2.$$

Since each call of $\textsc{Compare}(i, j, \varepsilon_l, \varepsilon_u, 2\delta/|S|^2)$ uses $O\Big(\frac{1}{(\varepsilon_u - \varepsilon_l)^2} \log \frac{|S|}{\delta}\Big)$ comparisons, bound on comparisons follows.

We now show that output is an ε_u-maximum. Let r^o be the output. We show that if $\tilde{p}_{e,r^o} > (\varepsilon_l + \varepsilon_u)/2$, then e is not eliminated before last round. Since $\tilde{p}_{e,r^o} > (\varepsilon_l + \varepsilon_u)/2$, by MST $\tilde{p}_{e,r^t} > (\varepsilon_l + \varepsilon_u)/2$ and hence not omitted by r^t (since $\textsc{Compare}(e, r^t, \varepsilon_l, \varepsilon_u, 2\delta/|S|^2)$ does not return 1). Hence all elements for which r^o is not $(\varepsilon_l + \varepsilon_u)/2$-preferable are present in the last round. Since anchor element is not updated in last round, $\textsc{Compare}(e, r^o, \varepsilon_l, \varepsilon_u, 2\delta/|S|^2)$ didn't return 3 for any remaining element and hence $\tilde{p}_{e,r^0} \leq \varepsilon_u$ for all elements in the last round and r^o

is an ε_u-maximum of S.

Further notice that either $r^o = r$ or $\tilde{p}_{r^o,r} > (\varepsilon_l + \varepsilon_u)/2$. □

4.C.4 Proof for Lemma 90

Proof. Let $m' = \frac{|S|}{m} \log \frac{|S|}{\delta}$. We now show that w.p.$\geq 1-\delta$, r is an (ε, m')-good anchor element. We prove this by showing that for every element e in S which is not an (ε, m')-*good anchor* element, Q will contain an element for which e is not ε-preferable. Let $e \in S$ be not an (ε, m')-*good anchor* element, then there are more than m' elements for which e is not ε-preferable. The probability that Q does not contain any element for which e is not ε-preferable is

$$\leq \left(1 - \frac{m'}{|S|}\right)^m = \left(1 - \frac{\log(|S|/\delta)}{m}\right)^m \leq \frac{\delta}{|S|}.$$

Let T be the set of all elements in S which are not (ε, m')-*good anchor* elements. Hence by union bound, w.p.$\geq 1-\delta$, for every element $e \in T$, Q has an element for which e is not ε-preferable.

If $r \in T$, then Q has an element for which r is not ε-preferable. But this contradicts our assumption that r is an ε-maximum of Q hence $r \notin T$. Therefore r is an (ε, m')-*good anchor* element. □

4.C.5 Proof for Lemma 54

Proof. Since $|Q| = \sqrt{|S| \log(4|S|/\delta)}$, by Corollary 52, SOFT-SEQ-ELIM$(Q, a, 0, \varepsilon/2, \delta/4)$ uses

$$O\left(\frac{|S| \log(4|S|/\delta)}{\varepsilon^2} \log \frac{|S| \log(4|S|/\delta)}{\delta}\right)$$
$$= O\left(\frac{|S|}{\varepsilon^2} \left(\log \frac{|S|}{\delta}\right)^2\right)$$

comparisons and w.p. $\geq 1 - \delta/4$, outputs r, an $\varepsilon/2$-maximum of Q.

By Lemma 90, w.p.$\geq 1-\delta/4$, r, an $\varepsilon/2$- maximum of Q is an $(\varepsilon/2, \sqrt{|S|\log(4|S|/\delta)})$-*good anchor* element of S.

Since r is an $(\varepsilon/2, \sqrt{|S|\log(4|S|/\delta)})$-*good anchor* element, by Lemma 51, w.p.$\geq 1-\delta/2$, SOFT-SEQ-ELIM$(S, r, \varepsilon/2, \varepsilon, \delta/2)$ uses

$$O\left(\frac{|S| + (\sqrt{|S|\log(4|S|/\delta)})^2}{\varepsilon^2}\log\frac{|S|}{\delta}\right)$$
$$= O\left(\frac{|S|}{\varepsilon^2}\left(\log\frac{|S|}{\delta}\right)^2\right)$$

comparisons and outputs an ε-maximum of S.

Proof follows from union bound. □

4.C.6 Proof of Lemma 55

Proof. Since Q contains $|S|^{3/4}$ elements, by Lemma 54, w.p.$\geq 1-\delta/3$, NEAR-OPT-MAX$(Q, \frac{\varepsilon}{2}, \frac{\delta}{3})$ outputs an $\frac{\varepsilon}{2}$-maximum of Q and uses

$$O\left(\frac{|S|^{3/4}}{\varepsilon^2}\left(\log\frac{|S|}{\delta}\right)^2\right) \overset{(a)}{=} O\left(\frac{|S|^{3/4}}{\varepsilon^2}\log\frac{|S|}{\delta}|S|^{1/4}\right)$$
$$\overset{(b)}{=} O\left(\frac{|S|}{\varepsilon^2}\log\frac{1}{\delta}\right)$$

comparisons where (a) is because $\delta \geq \min(1/|S|, e^{-|S|^{1/4}})$ and (b) is because $\delta \leq \frac{1}{|S|^{1/3}}$.

By Lemma 90, w.p.$\geq 1-\delta/3$, an $\varepsilon/2$-maximum of Q is an $\left(\frac{\varepsilon}{2}, \frac{|S|}{|S|^{3/4}}\log\frac{3|S|}{\delta}\right)$-*good anchor* element and hence r is an $\left(\varepsilon/2, |S|^{1/4}\log\frac{3|S|}{\delta}\right)$-*good anchor* element of S.

If r is an $\left(\varepsilon/2, |S|^{1/4}\log\frac{3|S|}{\delta}\right)$-*good anchor* element of S, by Lemma 51, w.p.$\geq 1-\delta/3$,

SOFT-SEQ-ELIM$(S, r, \frac{\varepsilon}{2}, \varepsilon, \frac{\delta}{4})$ outputs an ε-maximum of S and uses

$$\begin{aligned} & O\left(\frac{|S| + (|S|^{1/4}\log(3|S|/\delta))^2}{\varepsilon^2}\log\frac{|S|}{\delta}\right) \\ & \overset{(a)}{=} O\left(\frac{|S|}{\varepsilon^2}\log\frac{|S|}{\delta}\right) \\ & \overset{(b)}{=} O\left(\frac{|S|}{\varepsilon^2}\log\frac{1}{\delta}\right) \end{aligned}$$

comparisons where (a) is because $\delta \geq \min(1/|S|, e^{-|S|^{1/4}})$ and (b) is because $\delta \leq \frac{1}{n^{1/3}}$.

Result follows by union bound. □

4.C.7 COMPARE2

Algorithm 28 COMPARE2

inputs
  element i, element j, lower bias $\varepsilon_l \geq 0$, upper bias $\varepsilon_u > \varepsilon_l$, confidence δ
$\varepsilon_m \leftarrow (\varepsilon_l + \varepsilon_u)/2$, $\hat{\tilde{p}}_{i,j} \leftarrow 0$, $\hat{c} \leftarrow \frac{1}{2}$, $t \leftarrow 0$, $w \leftarrow 0$
while $|\hat{\tilde{p}}_{i,j} - \varepsilon_m| \leq \hat{c}$ and $t \leq \frac{2}{(\varepsilon_u - \varepsilon_l)^2}\log\frac{2}{\delta}$ **do**
  Compare i and j
  if i wins **then**
    $w \leftarrow w + 1$
  end if
  $t \leftarrow t + 1$
  $\hat{\tilde{p}}_{i,j} \leftarrow \frac{w}{t} - \frac{1}{2}$, $\hat{c} \leftarrow \sqrt{\frac{1}{2t}\log\frac{4t^2}{\delta}}$
end while
if $\hat{\tilde{p}}_{i,j} \leq \varepsilon_m$ **then**
  return 1
end if
return 2

Lemma 65. *For $\varepsilon_u > \varepsilon_l$,* COMPARE2$(i, j, \varepsilon_l, \varepsilon_u, \delta)$ *uses $\leq \frac{2}{(\varepsilon_u - \varepsilon_l)^2}\log\frac{2}{\delta}$ comparisons and if $\tilde{p}_{i,j} \leq \varepsilon_l$, then w.p. $\geq 1 - \delta$, it returns 1, else if $\tilde{p}_{i,j} \geq \varepsilon_u$, w.p. $\geq 1 - \delta$, it returns 2.*

4.C.8 High Ranges of Confidence

Notice that number of comparisons used by NEAR-OPT-MAX on a set of size $|S|/(\log|S|)^2$ is

$$O\left(\frac{|S|/(\log|S|)^2}{\varepsilon^2}\left(\log\frac{|S|}{\delta}\right)^2\right) = O\left(\frac{|S|}{\varepsilon^2}\right)$$

where last equality is because of $\delta \leq \frac{1}{|S|^{1/3}}$. So we can find an $\varepsilon/2$-maximum over a random set of size $|S|/(\log|S|)^2$ and use this element to prune the original set S and use SOFT-SEQ-ELIM over the pruned set. But it turns out we need multiple rounds of pruning and SOFT-SEQ-ELIM before we increase set size from $|S|/(\log|S|)^2$ to $|S|$. In each round we increase set size by a factor of $1/\delta$ until we reach the set size of $|S|$. We first present PRUNE that we use for pruning a set with an anchor element.

PRUNE

We do pruning similar to that in [FHO+17] but here we want to ensure that all better elements are still surviving not just the absolute maximum element. Hence we use similar pruning technique with small tweaks and derive different guarantees.

PRUNE takes five parameters: input set S, anchor element a, lower bias ε_l, upper bias ε_u and confidence δ. Goal of PRUNE is to output a set of size $\leq \frac{4\log(2|S|/\delta)}{\delta}$ (we later show that for our purpose this quantity is $O(|S|/\log|S|)$) such that it contains all elements for which a is not ε_u-preferable. If a is an (ε_l, n_1)-*good anchor* element, with $n_1 \leq \frac{2\log(2|S|/\delta)}{\delta}$, w.p.$\geq 1 - n_1\delta$, PRUNE$(S, a, \varepsilon_l, \varepsilon_u, \delta)$ achieves this goal using $O\left(\frac{|S|}{(\varepsilon_u-\varepsilon_l)^2}\log\frac{1}{\delta}\right)$ comparisons.

PRUNE prunes input set in rounds. At each round, it compares the remaining elements with the anchor element (sufficient number of times) and if it deems the element to be bad it eliminates the element. This ensures that number of bad elements decrease approximately by a factor of δ after each round. Notice that since number of elements decrease after each round,

PRUNE can afford to compare elements for more times in latter rounds for better accuracy. This process is continued until number of remaining elements is less than required target $\frac{4\log(2|S|/\delta)}{\delta}$.

For an anchor a, we call an element e as bad if $\tilde{p}_{e,a} \leq \varepsilon_l$ and good if $\tilde{p}_{e,a} \geq \varepsilon_u$. We want to ensure that number of bad elements decrease after each round and good elements never get eliminated. We can use COMPARE for this comparison. But we would like to mention that since requirement of comparison subroutine is less stringent than that required in SOFT-SEQ-ELIM, comparison subroutine COMPARE2 in [FHO+17] suffices here which in some cases can save a factor of 4 but has same orderwise complexity.

For completeness, we represent subroutine COMPARE2 in Appendix 4.C.7.

Algorithm 29 PRUNE

inputs
  Set S, element a, lower bias ε_l, upper bias ε_u, confidence δ.
$t \leftarrow 1$
$S_1 \leftarrow S$
while $|S_t| > 4\frac{\log\frac{2|S|}{\delta}}{\delta}$ and $t < \log^2 n$ **do**
  Initialize: $Q_t \leftarrow \emptyset$
  for e in S_t **do**
    if COMPARE2$(e, a, \varepsilon_l, \varepsilon_u, \delta/2^{t+1}) = 1$ **then**
      $Q_t \leftarrow Q_t \bigcup \{e\}$
    end if
  end for
  $S_{t+1} \leftarrow S_t \setminus Q_t$
  $t \leftarrow t+1$
end while
return S_t.

We now bound the number of comparisons used by PRUNE and prove its correctness.

Lemma 66. *If a is an (ε_l, n_1)-good anchor element with $n_1 \leq \frac{2\log(2|S|/\delta)}{\delta}$ then w.p.$\geq 1 - \frac{\delta}{2}$, PRUNE$(S, a, \varepsilon_l, \varepsilon_u, \delta)$ uses $O\Big(\frac{|S|}{(\varepsilon_u - \varepsilon_l)^2}\log\frac{1}{\delta}\Big)$ comparisons and outputs a set of size less than $\frac{4\log(2|S|/\delta)}{\delta}$. Further if a is not an ε_u-maximum of S then w.p.$\geq 1 - \frac{n_1\delta}{2}$, the output set contains all elements for which a is not ε_u- preferable.*

Proof. Proof is similar to proof of Lemma 5 in [FHO+17]. We present the proof for reader's convenience.

We first show that any element $e \in S$ for which a is not ε_u-preferable is part of output set w.p.$\geq 1-\delta/2$. e gets eliminated in round t if $\textsc{Compare2}(e,a,\varepsilon_l,\varepsilon_u,\delta/2^{t+1})$ returns 1 but since $\tilde{p}_{e,a} > \varepsilon_u$, w.p.$\geq 1-\delta/2^{t+1}$, $\textsc{Compare2}(e,a,\varepsilon_l,\varepsilon_u,\delta/2^{t+1})$ returns 2. Hence e gets eliminated in round t w.p.$\leq \delta/2^{t+1}$. Therefore by union bound, probability that e gets eliminated in any one of the rounds is $\leq \sum_t \delta/2^{t+1} \leq \delta/2$.

Since a is an (ε_l, n_1)-*good anchor* element number of elements for which a is not ε_u-preferable is $\leq n_1$. Hence invoking union bound once again, probability that any such element gets eliminated is $\leq n_1\delta/2$.

Now we bound output set size and number of comparisons used.

Notice that any element e for which a is ε_l-preferable if present in round t then gets eliminated in that round w.p.$\geq 1-\delta/2^{t+1}$, since $\textsc{Compare2}(e,a,\varepsilon_l,\varepsilon_u,\delta/2^{t+1})$ returns 1 with that probability. Hence if at the beginning of a round, the number of such elements (for which a is ε_l-preferable) is more than $\frac{2\log(2|S|/\delta)}{\delta}$, the probability that number of these elements surviving after round does not reduce by at least a factor of δ is

$$\begin{aligned}
\leq e^{-\frac{2\log(2|S|/\delta)}{\delta} D(\delta||\delta/2^{t+1})} &\leq e^{-\frac{2\log(2|S|/\delta)}{\delta} D(\delta||\delta/4)} \\
&\leq e^{-\frac{2\log(2|S|/\delta)}{\delta}\delta/2} \\
&= e^{-\log(2|S|/\delta)} \\
&= \frac{\delta}{2|S|}.
\end{aligned}$$

If number of elements for which a is ε_l-preferable decrease by a factor of δ after each round, then number of such elements fall below $\frac{2\log(2|S|/\delta)}{\delta}$ in $\leq \log_{1/\delta}\frac{|S|}{n_1}$ (since $n_1 \leq \frac{2\log(2|S|/\delta)}{\delta}$). Hence by union bound, w.p.$\geq 1-\delta/2$, number of such elements fall below $\frac{2\log(2|S|/\delta)}{\delta}$ in $\log_{1/\delta}\frac{|S|}{n_1}$ rounds. Henceforth we assume this and bound the number of comparisons.

Notice that number of elements for which a is not ε_l-preferable in round t is $\leq |S|\delta^{t-1}$. Since COMPARE2$(e,a,\varepsilon_l,\varepsilon_u,\delta')$ uses $\frac{2}{(\varepsilon_u-\varepsilon_l)^2}\log\frac{1}{\delta'}$ comparisons, total number of comparisons used on elements for which a is not ε_l-preferable is

$$\begin{aligned}
&\leq \sum_{t=1}^{\log_{1/\delta}\frac{|S|}{n_1}} \frac{2|S|\delta^{t-1}}{(\varepsilon_u-\varepsilon_l)^2}\log\frac{2^{t+1}}{\delta} \\
&\leq \frac{2|S|}{(\varepsilon_u-\varepsilon_l)^2}\sum_{t=1}^{\infty}\left(\delta^{t-1}\log\frac{1}{\delta}+(t+1)\delta^{t-1}\log 2\right) \\
&= O\left(\frac{|S|}{(\varepsilon_u-\varepsilon_l)^2}\log\frac{1}{\delta}\right)
\end{aligned}$$

where last equality follows since $\sum_{i=1}^{\infty}x^i$ and $\sum_{i}^{\infty}(i+1)x^{i-1}$ are bounded for $x\leq 1/2$.

We now bound the comparisons on elements for which a is not ε_l-preferable using number of rounds for which PRUNE runs. Number of comparisons used on elements for which a is not ε_l-preferable is

$$\begin{aligned}
&\leq \sum_{t=1}^{\log_{1/\delta}\frac{|S|}{n_1}} \frac{2n_1}{(\varepsilon_u-\varepsilon_l)^2}\log\frac{2^{t+1}}{\delta} \\
&\leq \frac{2n_1}{(\varepsilon_u-\varepsilon_l)^2}\sum_{t=1}^{\log_{1/\delta}\frac{|S|}{n_1}}\left(\log\frac{1}{\delta}+(t+1)\log 2\right) \\
&\leq \frac{2n_1}{(\varepsilon_u-\varepsilon_l)^2}\left(\left(\log_{1/\delta}\frac{|S|}{n_1}\right)\log\frac{1}{\delta}+\left(2\log_{1/\delta}\frac{|S|}{n_1}\right)^2\right) \\
&= O\left(\frac{|S|}{(\varepsilon_u-\varepsilon_l)^2}\log\frac{1}{\delta}\right).
\end{aligned}$$

Hence the Lemma follows. □

For better understanding, we further divide high confidence values into two ranges: 1) medium ($\frac{1}{\log|S|}\geq\delta\geq 1/|S|^{1/3}$) where one sequence of NEAR-OPT-MAX, PRUNE, SOFT-SEQ-ELIM is enough to produce output and 2) high ($\delta\geq\frac{1}{\log|S|}$) where NEAR-OPT-MAX and multiple sequences of PRUNE, SOFT-SEQ-ELIM are used to produce the final output.

OPT-MAX-MEDIUM

OPT-MAX-MEDIUM takes 3 parameters: input set S, bias ε and confidence δ such that $\frac{1}{\log|S|} \geq \delta \geq 1/|S|^{1/3}$. W.p.$\geq 1-\delta$, OPT-MAX-MEDIUM$(S,\varepsilon,\delta)$ uses $O\left(\frac{|S|}{\varepsilon^2}\log\frac{1}{\delta}\right)$ comparisons and outputs an ε-maximum of S..

OPT-MAX-MEDIUM picks a good anchor element and prunes the input set using this anchor element and use SOFT-SEQ-ELIM with the anchor on pruned set to output ε-maximum.

To pick the good anchor element it calls NEAR-OPT-MAX to find an $\varepsilon/3$-maximum of a random set of size $|S|/(\log|S|)^2$. We later show that this anchor is $\left(\varepsilon/3,(\log|S|)^2\log\frac{4|S|}{\delta}\right)$-*good anchor* element. Notice that elements for which anchor is not $\varepsilon/3$-preferable is $O(1/\delta^3)$. Hence using PRUNE with this anchor, we manage to prune set S to much smaller set Q'. Hence SOFT-SEQ-ELIM with this anchor on pruned set takes very few comparisons.

Algorithm 30 OPT-MAX-MEDIUM

inputs

  Set S, bias ε, confidence δ

Form a set Q by selecting $|S|/(\log|S|)^2$ random elements in S

$a \leftarrow$ NEAR-OPT-MAX$(Q,\varepsilon/3,\delta/4)$

$Q' \leftarrow$ PRUNE$\left(Q,a,\frac{\varepsilon}{3},\frac{2\varepsilon}{3},\frac{\delta}{4(\log|S|)^2\log(4|S|/\delta)}\right)$

return SOFT-SEQ-ELIM$(Q',a,\varepsilon/3,\varepsilon,\delta/4)$

In the below Lemma, we bound comparisons used by OPT-MAX-MEDIUM and prove its correctness.

Lemma 67. *For* $\frac{1}{\log|S|} \geq \delta \geq \frac{1}{|S|^{1/3}}$, *w.p.*$\geq 1-\delta$, OPT-MAX-MEDIUM$(S,\varepsilon,\delta)$ *uses* $O\left(\frac{|S|}{\varepsilon^2}\log\frac{1}{\delta}\right)$ *comparisons and outputs an* ε*-maximum of* S.

Proof. Since Q has $|S|/(\log|S|)^2$ elements, by Lemma 54, NEAR-OPT-MAX$(Q,\varepsilon/3,\delta/4)$ uses

$$O\left(\frac{|Q|}{\varepsilon^2}\left(\log\frac{|Q|}{\delta}\right)^2\right) = O\left(\frac{|S|}{\varepsilon^2(\log|S|)^2}\left(\log\frac{|S|}{\delta}\right)^2\right) = O\left(\frac{|S|}{\varepsilon^2}\right)$$

comparisons (where last equality is because $\delta \geq \frac{1}{|S|^{1/3}}$) and w.p.$\geq 1-\delta/3$ outputs a, an $\varepsilon/3$-maximum of Q.

By Lemma 90, w.p.$\geq 1-\delta/4$, an $\varepsilon/3$-maximum of Q is an $\left(\frac{\varepsilon}{3}, \frac{|S|}{|Q|}\log\frac{4|S|}{\delta}\right)$-*good anchor* element and hence a is an $\left(\frac{\varepsilon}{3}, (\log|S|)^2\log\frac{4|S|}{\delta}\right)$-*good anchor* element.

Let $\delta' = \frac{\delta}{4(\log|S|)^2\log(4|S|/\delta)}$. Notice that $\delta' \geq \frac{\delta}{16(\log|S|)^3}$. Since $(\log|S|)^2\log\frac{4|S|}{\delta} \leq \frac{2}{\delta'} < \frac{2\log(2|S|/\delta')}{\delta'}$, by Lemma 66, w.p.$\geq 1-\delta/4$, PRUNE$(S,a,\varepsilon/3,2\varepsilon/3,\delta'/4)$ uses

$$O\left(\frac{|S|}{\varepsilon^2}\log\frac{1}{\delta'}\right) = O\left(\frac{|S|}{\varepsilon^2}\log\frac{\log|S|}{\delta}\right) = O\left(\frac{|S|}{\varepsilon^2}\log\frac{1}{\delta}\right)$$

comparisons (where last equality is because $\delta \leq \frac{1}{\log|S|}$) and outputs a set Q' of size

$$\leq 4\frac{\log\frac{2|S|}{\delta'}}{\delta'} = O\left(\frac{(\log|S|)^4}{\delta}\right)$$

(where equality is because $\delta' \geq \frac{\delta}{16(\log|S|)^3}$ and $\delta \geq \frac{1}{|S|^{1/3}}$) s.t. Q' contains all elements in S for which a is not $2\varepsilon/3$-preferable i.e., $\tilde{p}_{e,a} \leq 2\varepsilon/3\ \forall e \in S\setminus Q'$.

W.p.$\geq 1-\delta/4$, SOFT-SEQ-ELIM$(Q',a,2\varepsilon/3,\varepsilon,\delta/4)$ uses

$$O\left(\frac{|Q'|^2}{\varepsilon^2}\log\frac{|Q'|}{\delta}\right) = O\left(\frac{(\log|S|)^8}{\delta^2\varepsilon^2}\log\frac{\log|S|}{\delta}\right) = O\left(\frac{|S|}{\varepsilon^2}\log\frac{1}{\delta}\right)$$

comparisons (where last equality is because $\frac{1}{\log|S|} \geq \delta \geq \frac{1}{|S|^{1/3}}$) and outputs r^o, an ε-maximum of

Q' s.t., either $r^o = a$ or $\tilde{p}_{r^o,a} \geq \frac{2\varepsilon}{3}$. Since $\tilde{p}_{e,a} \leq \frac{2\varepsilon}{3}\ \forall e \in S \setminus Q'$, by MST and Lemma 64, $\tilde{p}_{e,r^o} \leq \frac{2\varepsilon}{3}$ $\forall e \in S \setminus Q'$. Hence r^o is an ε-maximum of S.

Lemma follows by union bound and noting that comparisons used for each step is $O\left(\frac{|S|}{\varepsilon^2}\log\frac{1}{\delta}\right)$. □

As mentioned before for high ranges of confidence $\delta \geq 1/\log|S|$, our algorithm OPT-MAX-HIGH first uses NEAR-OPT-MAX over a set of size $|S|/(\log|S|)^2$ and then uses multiple rounds of PRUNE and SOFT-SEQ-ELIM before increasing size to $|S|$. After each round of PRUNE and SOFT-SEQ-ELIM, OPT-MAX-HIGH increases set size by a fraction of $1/\delta$.

OPT-MAX-HIGH

OPT-MAX-HIGH takes 3 parameters: input set S, bias ε and confidence δ such that $\delta \geq \frac{1}{\log|S|}$. W.p.$\geq 1-\delta$, OPT-MAX-HIGH$(S,\varepsilon,\delta)$ uses $O\left(\frac{|S|}{\varepsilon^2}\log\frac{1}{\delta}\right)$ comparisons and outputs an ε-maximum of S.

Similar to OPT-MAX-MEDIUM, to pick the initial anchor OPT-MAX-HIGH finds an $\varepsilon/3$-maximum of a random set of $|S|/(\log|S|)^2$ elements. We later show that this anchor is an $(\varepsilon/3, (\log|S|)^2\log(4|S|/\delta))$- *good anchor* element.

Notice that number of elements for which anchor is not $\varepsilon/3$-preferable could be much higher than $1/\delta^i$ for constant δ and any constant i. Hence a single round of pruning might not ensure that all such elements will survive. Hence we prune in stages improving anchor element in each stage using SOFT-SEQ-ELIM over pruned set in previous stage. After each stage, we improve anchor element and make sure that number of elements for which current anchor is not ε-preferable decreases.

In each stage we increase set size by a fraction of $1/\delta$ and prune the set using current anchor and use SOFT-SEQ-ELIM over pruned set with current anchor to find next anchor. Notice that initially set size is small and hence we can afford to repeat comparisons more times and thereby incur less bias error and confidence error in initial rounds. This makes sure that total bias

and confidence error are less than required.

Algorithm 31 OPT-MAX-HIGH

1: **inputs**

2: Set S, bias ε, confidence δ

3: Form a set Q by selecting $|S|/(\log|S|)^2$ random elements in S without replacement

4: $a_1 \leftarrow$ NEAR-OPT-MAX$(Q, \varepsilon/3, \delta/4)$

5: $m = \lceil 2\log_{1/\delta}\log|S| \rceil$

6: **for** i from 1 to m **do**

7: $n_i' \leftarrow \max(|S|/(\log|S|)^2, |S|\delta^{m-i})$

8: $\varepsilon_i' \leftarrow \varepsilon/3 + \frac{2\varepsilon/3}{2^{(m-i)/3}}$

9: $\varepsilon_i'' \leftarrow \varepsilon/3 + \frac{2\varepsilon/3}{2^{(m-i+1)/3}}$

10: $\delta_i' \leftarrow \delta^{m-i+4}$

11: Form a set Q_i by selecting n_i' random elements in S without replacement

12: $Q_i' \leftarrow$ PRUNE$(Q_i, a_i, \varepsilon_i'', (\varepsilon_i'' + \varepsilon_i')/2, \delta_i'^5/3)$

13: $a_{i+1} \leftarrow$ SOFT-SEQ-ELIM$(Q_i', a_i, \varepsilon_i'', \varepsilon_i', \delta_i'/3)$

14: **end for**

15: **return** a_{m+1}

In the below Lemma, we bound comparisons used by OPT-MAX-HIGH and prove its correctness.

Lemma 68. *For* $\delta \geq \frac{1}{\log|S|}$, *w.p.* $\geq 1-\delta$, OPT-MAX-HIGH(S, ε, δ) *uses* $O\left(\frac{|S|}{\varepsilon^2}\log\frac{1}{\delta}\right)$ *comparisons and outputs an* ε*-maximum of* S.

Proof. We first bound the comparisons used and confidence error accrued in base step 4

NEAR-OPT-MAX$(Q,\varepsilon/3,\delta/4)$. By Lemma 54, NEAR-OPT-MAX$(Q,\varepsilon/3,\delta/4)$ uses

$$O\left(\frac{|Q|}{\varepsilon^2}\left(\log\frac{|Q|}{\delta}\right)^2\right) = O\left(\frac{|S|}{\varepsilon^2(\log|S|)^2}\left(\log\frac{|S|}{\delta}\right)^2\right)$$
$$= O\left(\frac{|S|}{\varepsilon^2}\right)$$

comparisons (where last equality is because $\delta \geq \frac{1}{|S|^{1/3}}$) and w.p.$\geq 1-\delta/4$ outputs a_1, an $\varepsilon/3$-maximum of Q. Further by Lemma 90, w.p.$\geq 1-\delta/4$, , a_1 is an $\left(\varepsilon/3,(\log|S|)^2\log\frac{4|S|}{\delta}\right)$-*good anchor* element of S.

Hence by union bound, step 4 uses $O\left(\frac{|S|}{\varepsilon^2}\right)$ comparisons and w.p.$\geq 1-\delta/2$, outputs an $\left(\varepsilon/3,(\log|S|)^2\log\frac{4|S|}{\delta}\right)$-*good anchor* element of S.

From now we bound the comparisons used and confidence error incurred by steps 12, 13 in m recursions. We bound these quantities for a single recursion step i assuming that it got a correct answer from recursion step $i-1$.

Notice that at recursion step i, $n'_i = \max(|S|/(\log|S|)^2, |S|\delta^{m-i})$, $\varepsilon'_i = \varepsilon/3 + \frac{2\varepsilon/3}{2^{(m-i)/3}}$, $\varepsilon''_i = \varepsilon/3 + \frac{2\varepsilon/3}{2^{(m-i+1)/3}}$ and $\delta'_i = \delta^{m-i+4}$.

Assume that anchor element a_i at the start of recursion i is an $(\varepsilon''_i, 1/\delta'^4_i)$-*good anchor* element of S. Notice that assumption for recursion step 1 is true with probability $1-\delta/2$, since with same probability, a_1 is an $\left(\varepsilon/3,(\log|S|)^2\log\frac{4|S|}{\delta}\right)$-*good anchor* element of S. It is easy to check that $\varepsilon''_1 > \varepsilon/3$ and $1/\delta'^4_1 \geq (\log|S|)^2\log\frac{4|S|}{\delta}$. Hence a_1 is also an $(\varepsilon_1, 1/\delta'^4_1)$-*good anchor* element.

Now we will show that w.p.$\geq 1-\delta'_i$, recursion i uses $O\left(\frac{n'_i}{(\varepsilon'_i-\varepsilon''_i)^2}\log\frac{1}{\delta'_i}\right)$ comparisons and outputs an $(\varepsilon''_{i+1}, 1/\delta'^4_{i+1})$-*good anchor* element of S which will serve as an assumption for next recursion and that output is an ε'_i-maximum of set Q_i picked in step 11 which will show that at end of mth recursion we have an ε-maximum of S.

Let T_i be the set of all elements in S for which a_i is not ε''_i-preferable. Further add anchor element a_i to T_i i.e., $T_i = T_i \bigcup \{a_i\}$.

We now show that for any element e in T_i which is not $(\varepsilon_i', 1/\delta_{i+1}'^4)$-*good anchor* element, an element which is not ε_i'-preferable is present in Q_i and thereby later show that such an element won't be an output of ith recursion.

Notice that by assumption, $|T_i| \leq 1/\delta_i'^4 + 1$. Further let T_i' be the set of all elements in T_i which are not $(\varepsilon_i', 1/\delta_{i+1}'^4)$-*good anchor* elements. Observe that $|T_i'| \leq |T_i| \leq 1/\delta_i^4 + 1$.

Since in recursion i we pick set Q_i by picking n_i' elements randomly from S, each element in S is part of Q_i with probability $n_i'/|S|$.

For an element e in T_i', probability that Q_i does not contain an element for which e is not ε_i'-preferable is

$$\leq \left(1 - \frac{1/\delta_{i+1}'^4}{|S|}\right)^{n_i'} \leq \left(1 - \frac{(\delta/\delta_i')^4}{|S|}\right)^{|S|\delta_i'/\delta^4} \leq e^{-\frac{1}{\delta_i'^3}}$$

Hence by union bound, probability that for any element e in T_i', Q_i does not contain an element for which e is not ε_i'-preferable is

$$\leq |T_i'| e^{-\frac{1}{\delta_i'^3}} \leq \left(\frac{1}{\delta_i'^4} + 1\right) e^{-\frac{1}{\delta_i'^3}} \leq \delta_i'/3.$$

Now we bound comparisons used and confidence error incurred in step 12 during ith recursion.

Since a_i is an $(\varepsilon_i'', 1/\delta_i'^4)$-*good anchor* element and $1/\delta_i'^4 < 3/\delta_i'^5 < 2\frac{2\log(6|Q_i|/\delta_i'^5)}{\delta_i^5/3}$, by Lemma 66, $\textsc{Prune}(Q_i, a_i, \varepsilon_i'', (\varepsilon_i'' + \varepsilon_i')/2, \delta_i'^5/3)$ uses

$$O\left(\frac{|Q|}{(\varepsilon_i' - \varepsilon_i'')^2} \log \frac{1}{\delta_i'}\right) = O\left(\frac{n_i'}{(\varepsilon_i' - \varepsilon_i'')^2} \log \frac{1}{\delta_i'}\right)$$

comparisons and outputs a set Q_i' of size $\frac{12\log(6|Q_i|/\delta_i'^5)}{\delta_i'^5} = O\left(\frac{\sqrt{|Q_i|}}{\log|Q_i|}\right)$ (since $1/\delta_i' = O((\log|Q_i|)^5)$) and contains all elements in Q_i for which a_i is not an $(\varepsilon_i' + \varepsilon_i'')/2$-preferable i.e., $\tilde{p}_{e,a_i} \leq (\varepsilon_i'' + \varepsilon_i')/2$ $\forall e \in Q_i \setminus Q_i'$.

Now we bound comparisons used and confidence error incurred during step 13.

By Corollary 52, w.p.$\geq 1-\delta_i'/3$, SOFT-SEQ-ELIM$(Q_i', a_i, \varepsilon_i'', \varepsilon_i', \delta_i'/3)$ uses

$$O\left(\frac{|Q_i'|^2}{(\varepsilon_i''-\varepsilon_i')^2}\log\frac{|Q_i'|}{\delta_i'}\right) = O\left(\frac{|Q_i|/\log|Q_i|}{(\varepsilon_i'-\varepsilon_i'')^2}\log\frac{|Q_i|}{\delta_i'}\right)$$
$$= O\left(\frac{n_i'}{(\varepsilon_i'-\varepsilon_i'')^2}\log\frac{1}{\delta_i'}\right)$$

comparisons and outputs a_{i+1}, an ε_i'-maximum of Q_i' s.t., either $a_{i+1}=a_i$ or $\tilde{p}_{a_{i+1},a_i} > \frac{\varepsilon_i''+\varepsilon_i'}{2}$. Since $\tilde{p}_{e,a_i} \leq (\varepsilon_i''+\varepsilon_i')/2 \; \forall e \in Q_i \setminus Q_i'$, by MST and Lemma 64,$\tilde{p}_{e,a_{i+1}} \leq (\varepsilon_i''+\varepsilon_i')/2 \; \forall e \in Q_i \setminus Q_i'$. Hence a_{i+1} is an ε_i'-maximum of Q_i.

Further notice that since either $a_{i+1}=a_i$ or $\tilde{p}_{a_{i+1},a_i} > (\varepsilon_i'+\varepsilon_i'')/2 > \varepsilon_i''$, $a_{i+1} \in T_i$. Since for every $e \in T_i'$, Q_i contains an element for which e is not ε_i'-preferable, if $a_{i+1} \in T_i'$, it violates that a_{i+1} is an ε_i'-maximum of Q_i. Hence $a_{i+1} \notin T_i'$ and hence a_{i+1} is also an $(\varepsilon_i', (\delta/\delta_i')^4)$-*good anchor* element of S which is assumption for next recursion $i+1$.

Hence by union bound, w.p.$\geq 1-\delta_i$, recursion i uses $O\left(\frac{n_i'}{(\varepsilon_i'-\varepsilon_i'')^2}\log\frac{1}{\delta'}\right)$ comparisons and outputs an $(\varepsilon_i', 1/\delta_{i+1}'^4)$-*good anchor* element of S, which is also an ε_i'-maximum of Q_i.

Hence by union bound, total error incurred over all recursion steps and base case is

$$\leq \frac{\delta}{4} + \sum_{i=1}^{m}\delta_i' \leq \frac{\delta}{4} + \sum_{i=1}^{m}\delta^{m-i+4} \leq \frac{\delta}{4}+\frac{\delta}{2} < \delta.$$

Hence w.p.$\geq 1-\delta$, after m recursions, a_{m+1} is an ε-maximum of S.

Total number of comparisons used over all recursion steps and base case is

$$
\begin{aligned}
&= O\left(\frac{|S|}{\varepsilon^2}\right) + \sum_{i=1}^{m} O\left(\frac{n_i'}{(\varepsilon_i' - \varepsilon_i'')^2} \log \frac{1}{\delta_i'}\right) \\
&= \sum_{i=1}^{m} O\left(\frac{n_i' 2^{2(m-i)/3}}{\varepsilon^2} \log \frac{1}{\delta_i'}\right) \\
&= \sum_{i=1}^{m} O\left(\frac{|S|\delta^{m-i}2^{2(m-i)/3}}{\varepsilon^2} \log \frac{1}{\delta_i'}\right) \\
&\overset{(a)}{=} \sum_{i=1}^{m} O\left(\frac{|S|}{2^{(m-i)/3}\varepsilon^2} \log \frac{1}{\delta_i'}\right) \\
&= \sum_{i=1}^{m} O\left(\frac{|S|}{2^{(m-i)/3}\varepsilon^2} \log \frac{1}{\delta^{m-i+4}}\right) \\
&= O\left(\frac{|S|}{\varepsilon^2} \log \frac{1}{\delta}\right) \sum_{i=1}^{m} \frac{m-i+4}{2^{(m-i)/3}} \\
&= O\left(\frac{|S|}{\varepsilon^2} \log \frac{1}{\delta}\right)
\end{aligned}
$$

where (a) is because $\delta \leq 1/2$ and last equality because $x^i(i+4)$ series is convergent for $x \leq 1/2^{1/3}$. Hence proved. □

4.D Ranking

4.D.1 Outline of BINARY-SEARCH-RANKING and how to improve

[FOPS17] first proposes a ranking algorithm MERGE-RANK that uses $O\left(\frac{|S|(\log|S|)^3}{\varepsilon^2} \log \frac{|S|}{\delta}\right)$ comparisons and w.p.$\geq 1-\delta$ outputs an ε-ranking of S. They use this algorithm as a building block for the final algorithm BINARY-SEARCH-RANKING. We outline BINARY-SEARCH-RANKING algorithm and mention components that lead to additional $(\log\log n)^3$ factor. We also propose modified components that remove additional $(\log\log n)^3$ factor.

BINARY-SEARCH-RANKING first selects $\frac{|S|}{(\log|S|)^3}$ random elements in S as anchors and ranks them using MERGE-RANK. Notice that this step uses $O\left(\frac{|S|\log|S|}{\varepsilon^2}\right)$ comparisons.

Then BINARY-SEARCH-RANKING forms bins between two consecutively ranked anchors. For each element e BINARY-SEARCH-RANKING finds which bin it belongs to using INTERVAL-BINARY-SEARCH. INTERVAL-BINARY-SEARCH does a noisy random walk over the bins to determine which bin element e belongs to. Noisy random walk is run for $O(\log|S|)$ steps where in each step $O(\frac{1}{\varepsilon^2})$ comparisons are used. Notice that this random walk step uses $O(\frac{\log|S|}{\varepsilon^2})$ comparisons for each element e and hence $O(\frac{|S|\log|S|}{\varepsilon^2})$ comparisons over all elements. This noisy random walk can sometimes fail if element e visits a close anchor a, i.e., $|\tilde{p}_{e,a}| \leq k\varepsilon$ for some constant $k < 1$.

Then INTERVAL-BINARY-SEARCH uses BINARY-SEARCH over visited anchors to find closeby anchor. Notice that number of visited anchors during noisy random walk is $O(\log|S|)$. BINARY-SEARCH does a binary search over ranked set of visited anchors and hence runs for $O(\log\log|S|)$ steps. It uses $O\left(\frac{\log|S|}{\varepsilon^2}\right)$ comparisons over each step to ensure that error probability during is $1/|S|^9$ for each element. Hence BINARY-SEARCH uses $O(\frac{\log|S|(\log\log|S|)}{\varepsilon^2})$ comparisons for each element and therefore uses $O(\frac{|S|\log|S|(\log\log|S|)}{\varepsilon^2})$ comparisons over all elements incurring an overhead factor of $\log\log|S|$.

We fix this step BINARY-SEARCH by not using $O(\log|S|)$ comparisons over each step but instead $O(\log\log|S|)$ comparisons over each steps. Notice that this increases error probability to $1/\log|S|$ for each element in place of $1/|S|^9$. Hence we check if in fact BINARY-SEARCH found a closeby anchor by comparing with the output anchor element for $O(\log|S|)$ times which will give a wrong decision only w.p. $\leq 1/|S|^{10}$. If we find that output anchor element is indeed closeby anchor then we output it or else this time we do one more round of binary search repeating each comparison $O(\log|S|)$ times thereby ensuring that final output is correct w.p.$\geq 1-\frac{1}{|S|^9}$.

Notice that even though worst case complexity for each element is same as before, most of the elements use much fewer comparisons. Since first round of binary search is correct w.p.$\geq 1-1/\log|S|$, much fewer than $10/\log|S|$ fraction of elements fail to find closeby anchor during first round. Hence less than $10/\log|S|$ fraction of total elements use second round and

hence total comparisons are $O(\frac{|S|\log|S|}{\varepsilon^2})$ comparisons.

Later for each bin, BINARY-SEARCH-RANKING finds if elements are close to bin's boundary anchor elements by repeating comparisons $O(\log|S|)$ times and if an element is close to boundary anchor it ranks that element close to boundary anchor. Notice that this step uses $O(\frac{\log|S|}{\varepsilon^2})$ comparisons for each element and hence $O(\frac{|S|\log|S|}{\varepsilon^2})$ comparisons over all elements. Therefore this step is not a problem.

It can be shown that after removing elements which are close to bin boundaries each bin has $O((\log|S|)^4)$ elements.

Then BINARY-SEARCH-RANKING uses MARGE-RANK to rank each bin with error probability $1/|S|^3$ for each bin. Hence it uses $O(\frac{|B_i|(\log|B_i|)^3}{\varepsilon^2}\log\frac{|B_i|}{1/|S|^3}) = O(\frac{|B_i|(\log|B_i|)^3}{\varepsilon^2}\log|S|) = O(\frac{|B_i|(\log\log|S|)^3}{\varepsilon^2}\log|S|)$ comparisons (since $|B_i| = O((\log n)^4)$) for each bin B_i. When we sum over all bins B_i, total comparisons is $O(\frac{|S|\log|S|(\log\log|S|)^3}{\varepsilon^2})$ which has an overhead factor of $(\log\log|S|)^3$.

We improve this step by not using error probability of $1/|S|^3$ but using $1/\log|S|$ for error probability. This ensures that comparisons are bounded but overall error probability is high. So we check if each bin is ranked correctly. Our main contribution is an algorithm to find an ordered set is ε-ranked. Using this algorithm we find if each bin is correctly ranked by making sure that we know right answer w.p.$\geq 1-\frac{1}{|S|^5}$. Checking over all bins with this algorithm uses only $O(\frac{|S|\log|S|}{\varepsilon^2})$ comparisons. If a bin is not correctly ranked, then we rank that bin again this time with error probability of $1/|S|^4$.

Observe that in worst case, ranking a bin uses same comparisons as previously but only few bins need these many comparisons since most of the bins are ranked correctly in first round. Notice that since first round of ranking bin is correct with probability $\geq 1-\frac{1}{\log|S|}$, $< 10/\log|S|$ fraction of bins need second round of ranking which helps us in bounding comparisons.

We first present new BINARY-SEARCH that has low overall sample complexity than the one in [FOPS17]. We prove Lemmas similar to those in [FOPS17] with lower complexities.

4.D.2 BINARY-SEARCH

BINARY-SEARCH uses a subroutine BUD-BINARY-SEARCH that takes a confidence parameter δ and outputs desired result w.p.$\geq 1-\delta$.

BUD-BINARY-SEARCH

As outlined previously, BUD-BINARY-SEARCH does a simple binary search using $O\left(\frac{1}{\varepsilon^2}\log\frac{\log|Q|}{\delta}\right)$ comparisons (where Q is a set over which binary search is performed) for each step.

Algorithm 32 BUD-BINARY-SEARCH

inputs
  Ordered array S, ordered array Q, search item e, bias ε, confidence δ
$l \leftarrow 1, h \leftarrow |Q|$.
while $h-l>0$ **do**
  Compare e and $S(Q(\lceil\frac{l+h}{2}\rceil))$ for $\frac{4}{\varepsilon^2}\log\frac{10\log|Q|}{\delta}$ times.
  $t \leftarrow$ fraction of times e won.
  if $t \in \left[\frac{1}{2}-3\varepsilon, \frac{1}{2}+3\varepsilon\right]$ **then**
    return $Q(\lceil\frac{l+h}{2}\rceil)$.
  else if $t < \frac{1}{2}-3\varepsilon$ **then**
    $h = \lceil\frac{l+h}{2}\rceil$
  else
    $l = \lceil\frac{l+h}{2}\rceil$
  end if
end while
return $Q(h)$.

Lemma 69. *For $\varepsilon'' > \varepsilon'$, consider ordered sets S, Q s.t. $\tilde{p}_{S(Q(i)),S(Q(j))} \leq \varepsilon' \forall i < j$. For an*

element e s.t., $\exists g : |\tilde{p}_{S(Q(g)),a}| < 2\varepsilon''$, BINARY-SEARCH$(S, Q, a, \varepsilon'', \delta)$ *uses* $O(\frac{\log|Q|}{\varepsilon''^2}\log\frac{\log|Q|}{\delta})$ *comparisons and w.p.*$\geq 1-\delta$ *returns y s.t.* $|\tilde{p}_{S(Q(y)),a}| < 4\varepsilon''$.

Proof. This proof is similar to Lemma 23 in [FOPS17]

Notice that this is a binary search over ordered set Q using $O(\frac{1}{\varepsilon''^2}\log\frac{\log|Q|}{\delta})$ comparisons for each step and hence bound on comparisons follow.

At any stage of BUD-BINARY-SEARCH, there are three possibilities that can happen . Consider the case when we are comparing e with $S(Q(i))$.

1. $|\tilde{p}_{S(Q(i)),e}| < 2\varepsilon''$. Probability that the fraction of wins for e is not between $\frac{1}{2}-3\varepsilon''$ and $\frac{1}{2}+3\varepsilon''$ is less than $e^{-\frac{\log(4|Q|/\delta)}{\varepsilon''^2}\varepsilon''^2} \leq \frac{\delta}{4\log|Q|}$. Hence BUD-BINARY-SEARCH outputs $Q(i)$.

2. $\tilde{p}_{S(Q(i)),e} > 2\varepsilon''$. Probability that the fraction of wins for e is more than $\frac{1}{2}$ is less than $e^{-\frac{\log(4|Q|/\delta)}{\varepsilon''^2}\varepsilon''^2} \leq \frac{\delta}{4\log|Q|}$. So BUD-BINARY-SEARCH will not move right. Also notice that $\tilde{p}_{S(Q(j)),e} > 2\varepsilon'' - \varepsilon' > \varepsilon'' \; \forall j > i$.

3. $\tilde{p}_{S(Q(i)),e} > 4\varepsilon''$. Probability that the fraction of wins for e is more than $\frac{1}{2}-3\varepsilon''$ is less than $e^{-\frac{\log(4|Q|/\delta)}{\varepsilon''^2}\varepsilon''^2} \leq \frac{\delta}{4\log|Q|}$. Hence BUD-BINARY-SEARCH will move left. Also notice that $\tilde{p}_{S(Q(j)),e} > 4\varepsilon'' - \varepsilon' > \varepsilon'' \; \forall j > i$.

We can show similar results for $\tilde{p}_{S(Q(i)),e} < -2\varepsilon''$ and $\tilde{p}_{S(Q(i)),e} < -4\varepsilon''$. Hence if $|\tilde{p}_{S(Q(i)),e})| < 2\varepsilon''$ then BUD-BINARY-SEARCH outputs $Q(i)$, and if $2\varepsilon'' < |\tilde{p}_{S(Q(i)),e}| < 4\varepsilon''$ then either BUD-BINARY-SEARCH outputs $Q(i)$ or moves in the correct direction and if $|\tilde{p}_{S(Q(i)),e}| > 4\varepsilon''$, then BUD-BINARY-SEARCH moves in the correct direction.

Proof follows by noting that that binary search visits $\log|Q|$ indices and using union bound. □

BINARY-SEARCH

BINARY-SEARCH initially calls BUD-BINARY-SEARCH with confidence parameter of $1/(4\log|S|)$ and then it checks if BUD-BINARY-SEARCH gave a required result by comparing with output for $O(\frac{\log|S|}{\varepsilon^2})$ times. If it deems that output is not good then it calls BUD-BINARY-

SEARCH this time with confidence parameter $1/|S|^{10}$. Notice that if BINARY-SEARCH is called $|S|$ independent times then less than $10/\log|S|$ fraction of times second BUD-BINARY-SEARCH is called since first BUD-BINARY-SEARCH gives required answer w.p.$\geq 1-1/(4\log|S|)$.

Algorithm 33 BINARY-SEARCH

inputs
  Ordered Set S, ordered array Q, search item e, bias ε
$l \leftarrow$ BUD-BINARY-SEARCH$(S,Q,e,\varepsilon,1/4\log|S|)$
Compare e and $S(l)$ for $\frac{20\log|S|}{\varepsilon^2}$ times. $t \leftarrow$ fraction of times e won
if $t \in [1/2-5\varepsilon, 1/2+5\varepsilon]$ **then**
  return l
else
  return BUD-BINARY-SEARCH$(S,Q,e,\varepsilon,1/|S|^{10})$
end if

Lemma 70. *For $\varepsilon'' > \varepsilon'$, consider ordered sets S, Q s.t. $\tilde{p}_{S(Q(i)),S(Q(j))} \leq \varepsilon' \forall i < j$ and $|Q| \leq 90\log|S|$. For an element e s.t., $\exists g : |\tilde{p}_{S(Q(g)),a}| < 2\varepsilon''$,* BINARY-SEARCH$(S,Q,a,\varepsilon'',\delta)$ *w.p. $\geq 1-1/|S|^9$ returns y s.t. $|\tilde{p}_{S(Q(y)),a}| < 6\varepsilon''$ and w.p. $\geq 1-1/(3\log|S|)$ uses $O\left(\frac{\log|S|}{\varepsilon^2}\right)$ comparisons and always uses $O\left(\frac{(\log|S|)(\log\log|S|)}{\varepsilon^2}\right)$ comparisons. If repeated for $|S|$ independent copies of Q and e, w.p.$\geq 1-1/|S|^9$,* BINARY-SEARCH *uses $O\left(\frac{|S|\log|S|}{\varepsilon^2}\right)$ comparisons.*

Proof. BUD-BINARY-SEARCH$(S,Q,e,\varepsilon,1/(4\log|S|))$ uses $O(\frac{\log\log|S|}{\varepsilon^2}\log\log|S|)$ comparisons and w.p.$\geq 1-1/(4\log|S|)$ outputs y s.t. $|\tilde{p}_{S(Q(y)),a}| < 4\varepsilon''$.

If $|\tilde{p}_{S(Q(y)),e}| < 4\varepsilon''$, probability that the fraction of wins for e is not between $1/2-5\varepsilon''$ and $1/2+5\varepsilon''$ is less than $e^{-\frac{20\log|S|}{\varepsilon''^2}\varepsilon''^2} \leq \frac{1}{|S|^{20}}$.

If first BUD-BINARY-SEARCH fails that is if $|\tilde{p}_{S(Q(y)),e}| > 6\varepsilon''$, probability that the fraction of wins for e is between $1/2-5\varepsilon''$ and $1/2+5\varepsilon''$ is less than $e^{-\frac{20\log|S|}{\varepsilon''^2}\varepsilon''^2} \leq \frac{1}{|S|^{20}}$ and hence second BUD-BINARY-SEARCH is invoked.

BUD-BINARY-SEARCH$(S,Q,e,\varepsilon,1/|S|^{10})$ uses $O(\frac{\log\log|S|}{\varepsilon^2}\log|S|)$ comparisons and w.p.$\geq 1-\frac{1}{|S|^{10}}$ outputs y s.t. $|\tilde{p}_{S(Q(y)),a}|<4\varepsilon''$.

So by union bound, w.p.$\geq 1-1/|S|^9$, BINARY-SEARCH outputs required answer and w.p.$\geq 1-\frac{1}{3\log|S|}$ uses $O(\frac{|S|\log|S|}{\varepsilon^2})$ comparisons and always uses $O(\frac{(\log|S|)(\log\log|S|)}{\varepsilon''^2})$ comparisons.

If we repeat this for $|S|$ independent copies, then probability that second BUD-BINARY-SEARCH used for more than $20/|\log|S|$ is less than $e^{-|S|D(\frac{10}{\log|S|}||\frac{1}{3\log|S|})}\leq e^{-10|S|/\log|S|}\leq 1/|S|^9$. Hence the bound on comparisons follows. □

We now present RANK-CHECK that checks if an ordered set is ε-ranked or not 3ε-ranked.

4.D.3 RANK-CHECK

RANK-CHECK takes three parameters: ordered set S, bias ε and confidence δ. RANK-CHECK deems if S is ε-ranked i.e., $\tilde{p}_{S(i),S(j)}\leq\varepsilon\ \forall i<j$ or if S is not 3ε-ranked i.e., there exists $i<j$ s.t. $\tilde{p}_{S(i),S(j)}>3\varepsilon$. RANK-CHECK$(S,\varepsilon,\delta)$ uses $O\left(\frac{|S|}{\varepsilon^2}\log\frac{|S|}{\delta}\right)$ comparisons and w.p.$\geq 1-\delta$, outputs a correct decision.

RANK-CHECK first sets anchor a as $S(1)$ and iterates over the set S checking if $\tilde{p}_{S(i),a}>\varepsilon$ vs $\tilde{p}_{S(i),a}<0$ using COMPARE2. If COMPARE2 deems $\tilde{p}_{S(i),a}>\varepsilon$, then RANK-CHECK updates current anchor to $S(i)$. If COMPARE2 deems $\tilde{p}_{S(i),a}<0$, then RANK-CHECK checks if $\tilde{p}_{a,S(i)}<\varepsilon$ vs $\tilde{p}_{a,S(i)}>2\varepsilon$ again using COMPARE2. If COMPARE2 deems $\tilde{p}_{a,S(i)}>2\varepsilon$, then $S(i)$ is much worse than a which is to the left of $S(i)$ in S and hence S is not ε-ranked and RANK-CHECK returns FALSE.

Due to probabilistic nature if S is neither ε-ranked nor not 3ε-ranked then it can either of the two outputs.

Algorithm 34 RANK-CHECK

inputs
  Ordered Set S, bias ε, confidence δ
$a \leftarrow S(1)$
for i from 2 to $|S|$ **do**
  if COMPARE2$(S(i), a, 0, \varepsilon, \frac{\delta}{2|S|}) = 2$ **then**
    $a \leftarrow S(i)$
  else if COMPARE2$(a, S(i), \varepsilon, 2\varepsilon, \frac{\delta}{2|S|}) = 2$ **then**
    return FALSE
  end if
end for
return TRUE

In the below Lemma, we bound comparisons used by RANK-CHECK and prove its correctness.

Lemma 71. *For an ordered set S,* RANK-CHECK(S, ε, δ) *uses* $O\left(\frac{|S|}{\varepsilon^2} \log \frac{|S|}{\delta}\right)$ *comparisons. If S is an* ε*-ranked set i.e.,* $\tilde{p}_{S(i),S(j)} \leq \varepsilon \; \forall i < j$*, w.p.*$\geq 1 - \delta$*,* RANK-CHECK$(S, \varepsilon, \delta)$ *outputs TRUE. If S is not a* 3ε*-ranked set i.e.,* $\exists i < j$ *s.t.* $\tilde{p}_{S(i),S(j)} > 3\varepsilon$ *then w.p.*$\geq 1 - \delta$*,* RANK-CHECK(S, ε, δ) *outputs FALSE.*

Proof. We first bound the comparisons.

Since both COMPARE2$(e, f, 0, \varepsilon, \delta/(2|S|))$ and COMPARE2$(e, f, \varepsilon, 2\varepsilon, \delta/(2|S|))$ use $O(\frac{1}{\varepsilon^2} \log \frac{|S|}{\delta})$ comparisons and each one is called for at most $|S|$ times, total number of comparisons used is $O\left(\frac{|S|}{\varepsilon^2} \log \frac{|S|}{\delta}\right)$.

Since each COMPARE2 is called with confidence parameter of $\delta/(2|S|)$ and total number of COMPARE2 calls are less than $2|S|$, by union bound, w.p.$\geq 1 - \delta$, all COMPARE2 calls give desired result i.e., COMPARE2$(e, f, \varepsilon_l, \varepsilon_u, \delta/(2|S|))$ outputs 1 if $\tilde{p}_{e,f} \leq \varepsilon_l$ and 2 if $\tilde{p}_{e,f} \geq \varepsilon_u$.

Let a_j be the anchor element a that is compared with $S(j)$. Notice that $a_2 = S(1)$.

We first consider the case of ε-ranked set S. Notice that since anchor element is always left to competing element in the set S i.e., $a_j = S(i)$ for some $i < j$. Since $\tilde{p}_{S(i),S(j)} \leq \varepsilon$, $\forall i < j$, $\tilde{p}_{a_j,S(j)} \leq \varepsilon$ and hence COMPARE2$(a_j, S(j), \varepsilon, 2\varepsilon, \delta/(2|S|))$ outputs 1 for all j. Therefore FALSE is never returned.

Now we consider the second case: S is not an 3ε-ranked set. Notice that there exists $i < j$ s.t. $\tilde{p}_{S(i),S(j)} > 3\varepsilon$. If RANK-CHECK reaches $S(j)$, then we show that $\tilde{p}_{a_j,S(j)} > 2\varepsilon$ thereby proving the required result.

We first show that anchor element only keeps getting better i.e., $\tilde{p}_{a_l,a_k} \geq 0 \; \forall l \geq k$. Since COMPARE2$(S(k), a_k, 0, \varepsilon, \delta/(2|S|))$ does not output 2 if $\tilde{p}_{S(k),a_k} < 0$, $a_{k+1} = S(k)$ only if $\tilde{p}_{S(k),a_k} \geq 0$. Hence $\tilde{p}_{a_{k+1},a_k} \geq 0$. Therefore by SST, $\tilde{p}_{a_l,a_k} \geq 0 \; \forall l \geq k$.

We now show that $\tilde{p}_{S(k),a_{k+1}} < \varepsilon$. Notice that $a_{k+1} = a_k$ if COMPARE2$(S(k), a_k, 0, \varepsilon, \delta/(2|S|))$ outputs 1 and hence $\tilde{p}_{S(k),a_k} < \varepsilon$. Hence either $a_{k+1} = S(k)$ or $a_{k+1} = a_k$ and $\tilde{p}_{S(k),a_{k+1}} < \varepsilon$. Therefore $\tilde{p}_{S(k),a_{k+1}} < \varepsilon$.

Since $\tilde{p}_{S(i),a_{i+1}} < \varepsilon$ and $\tilde{p}_{S(i),S(j)} > 3\varepsilon$, by SST and STI, $\tilde{p}_{a_{i+1},S(j)} > 2\varepsilon$. Because of $\tilde{p}_{a_j,a_{i+1}} \geq 0$, by SST, $\tilde{p}_{a_j,S(j)} > 2\varepsilon$. Hence COMPARE2$(S(j), a_j, 0, \varepsilon, \delta/(2|S|))$ outputs 1 and COMPARE2$(a, S(i), \varepsilon, 2\varepsilon, \delta/2|S|)$ outputs 2 thereby resulting in FALSE output. □

We now briefly talk about how to use RANK-CHECK to improve sample complexity of ranking bins

4.D.4 Ranking Bins

Similar to modified BINARY-SEARCH, we first rank each bin using MERGE-RANK with bias parameter of $\varepsilon/3$ and confidence parameter of $1/(4\log|S|)$ and check if bin is correctly ranked using RANK-CHECK with confidence parameter of $1/|S|^{10}$. Notice that RANK-CHECK uses only $O(\frac{|B_i|}{\varepsilon^2}\log|S|)$ comparisons for each bin B_i. If RANK-CHECK deems that bin is not

correctly ranked then we use a second round of MERGE-RANK with bias parameter of ε and confidence parameter of $1/|S|^{10}$.

Notice that since we rank $|S|/(\log|S|)^3$ bins, less than $20/\log|S|$ fraction of them use second round of MERGE-RANK and hence total number of comparisons is $\sum_i O(\frac{|B_i|}{\varepsilon^2}\log|S|) = O(\frac{|S|\log|S|}{\varepsilon^2})$. Proof is similar to that of Lemma 70.

4.D.5 Final Ranking Algorithm

Use BINARY-SEARCH-RANKING presented in [FOPS17] with new BINARY-SEARCH subroutine in place of the one presented there. Also replace 5(b) step there which ranks each bin in one go with a round of ranking followed by checking with RANK-CHECK if ranking is done correctly and repeating ranking if not done correctly.

Hence we improved both components that were incurring additional $\log\log|S|$ factors and thereby removed $(\log\log|S|)^3$ factor.

4.E Proof for Theorem 58

Proof. This proof is similar to proof of Theorem 7 in [FHO$^+$17].

Consider the model where $\tilde{p}_{a_1,a_2} = 1/2$, $\tilde{p}_{a_i,a_j} = (0 <)\mu(\ll 1/n^{10})$, when $i < j$ and $(i,j) \neq (1,2)$. This model has an order: $a_1 \succ a_2 \succ \cdots \succ a_{n-1} \succ a_n$ i.e., $\tilde{p}_{a_i,a_j} > 0\ \forall i < j$. Further this model satisfies MST since $\tilde{p}_{a_i,a_k} \geq \min(\tilde{p}_{a_i,a_j}, \tilde{p}_{a_j,a_k})\ \forall i < j < k$. This model also satisfies STI since $\tilde{p}_{a_i,a_k} \leq \tilde{p}_{a_i,a_j} + \tilde{p}_{a_j,a_k}\ \forall i < j < k$.

We prove the Lemma by reducing the above model to the model where μ is replaced by 0.

Note that μ is so small that if we consider a model where we replace μ with 0, the comparisons behave essentially similarly. More formally, let model M_μ be the model we consider and M_0 be the model when μ is replaced with 0. Let C denote a sequence of comparisons where each element of the sequence includes the elements compared and its outcome. Further, for each

sequence C, let $P_\mu(C)$ and $P_0(C)$ denote the probability of sequence C under models M_μ and M_0 respectively. Now consider a sequence C of comparisons of length $\leq n^2/20$ (chosen to make proof simpler. One can also prove for constants higher than $1/20$). Then

$$\frac{P_0(C)}{P_\mu(C)} \geq \left(\frac{1/2}{1/2+\mu}\right)^{n^2/20} \geq e^{-n^2/(10n^{10})} \geq \frac{6}{7}$$

Thus the probability of any sequence of length $\leq \frac{n^2}{20}$ is approximately same under both models. Hence if there is an algorithm that uses $\leq \frac{n^2}{20}$ comparisons and w.p.$\geq 7/8$ produces an $1/4$-ranking under M_μ model then applying same algorithm over M_0 model produces an $1/4$-ranking w.p.$\geq \frac{7}{8} \cdot \frac{6}{7} = \frac{3}{4}$.

We now show that there exists no algorithm that uses $\leq \frac{n^2}{20}$ comparisons and w.p.$\geq \frac{3}{4}$ generates a $1/4$-ranking under M_0, thus proving the Lemma. It is easy to see that any ordering outputted without querying the comparison between a_1 and a_2 is a $1/4$-ranking w.p. exactly 1/2 since no order between a_1 and a_2 can be deduced. Since the pair (a_1, a_2) is one random pair among $\binom{n}{2}$ pairs, the probability that the algorithm asks a comparison between this pair with $n^2/20$ comparisons is $< \frac{1}{2}$. So the probability that the output order contains a_1 and a_2 in the right order is $< \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} = \frac{3}{4}$. □

4.F Approximating Pairwise Probabilties

4.F.1 Proof for Theorem 59

Proof. Consider the model $\{a_1, a_2, ..., a_n\}$ where $(4+4k)\varepsilon \leq \tilde{p}_{a_{i+k},a_i} \leq (8+4k)\varepsilon$ for $1 \leq k \leq \min(n-i, \lfloor \frac{1}{16\varepsilon} - 2 \rfloor)$ and $\tilde{p}_{a_{i+k},a_i} = 1/4$ for $k > \min(n-i, \lfloor \frac{1}{16\varepsilon} - 2 \rfloor)$.

Notice that this model satisfies SST since there exists an ordering $\succ$ i.e., $a_j \succ a_i$ if $j > i$ (since $\tilde{p}_{a_j,a_i} > 0$) and $\tilde{p}_{a_k,a_i} \geq \min(\tilde{p}_{a_k,a_j}, \tilde{p}_{a_j,a_i})$ $\forall k > j > i$.

Further observe that this model also satisfies STI since $\tilde{p}_{a_k,a_i} \leq \tilde{p}_{a_k,a_j} + \tilde{p}_{a_j,a_i}$ $\forall k > j > i$.

Notice that even after knowing every other pairwise probability other than for pair $\{a_k, a_i\}$ exactly, SST and STI do not impose any additional constraints on $\tilde{p}_{a_k,a_i}$ than those already given by the model definition. Hence we can only infer $\tilde{p}_{a_k,a_i}$ from model definition and comparisons between a_k and a_i.

Model does not fix pairwise probability for

$$\geq \sum_{i=1}^{n} \min\left(n-i, \frac{1}{16\varepsilon} - 2\right) = \Omega(n \min(n, 1/\varepsilon))$$

pairs and for each such pair $\{a_i, a_j\}$, $\tilde{p}_{a_i,a_j}$ is known only upto an accuracy of 2ε.

So any algorithm has to approximate $\Omega(n\min(n,1/\varepsilon))$ probabilities between $1/2$ and $3/4$ to an accuracy of ε. Using Information Theoretic arguments, it follows that any algorithm needs $\Omega\left(\frac{n\min(n,1/\varepsilon)}{\varepsilon^2}\log n\right)$ comparisons for error accuracy of $\leq 1/4$. □

4.F.2 Properties of ε-ranked ordered Set

We first prove some properties of ε-ranked ordered set that we use in proving correctness of APPROX-PROB.

Lemma 72. *If S is an ε-ranked ordered set i.e., $\tilde{p}_{S(i),S(j)} \leq \varepsilon \ \forall i < j$, then $\tilde{p}_{S(k),S(j)} \leq \tilde{p}_{S(k),S(i)} + \varepsilon$ $\forall k \geq j \geq i$ and $\tilde{p}_{S(k),S(i)} \geq \tilde{p}_{S(j),S(i)} - \varepsilon \ \forall k \geq j \geq i$.*

Proof. Notice that if $S(j) \succ S(i)$, then by SST $\tilde{p}_{S(k),S(j)} \leq \tilde{p}_{S(k),S(i)}$. If $S(i) \succ S(j)$, then since S is ε-ranked, $\tilde{p}_{S(i),S(j)} \leq \varepsilon$, and hence by SST and STI, $\tilde{p}_{S(k),S(j)} \leq \tilde{p}_{S(k),S(i)} + \tilde{p}_{S(i),S(j)} \leq \tilde{p}_{S(k),S(i)} + \varepsilon$.

Similarly it can be shown that $\tilde{p}_{S(k),S(i)} \geq \tilde{p}_{S(j),S(i)} - \varepsilon \ \forall k \geq j \geq i$. □

4.F.3 Proof for Theorem 60

Proof. We first prove the correctness. Let $\varepsilon' = \varepsilon/8$. Notice that S is ε'-ranked. Let $\hat{\tilde{p}}_{S(i),S(j)}$ be the approximated output of $\tilde{p}_{S(i),S(j)}$.

Observe that whenever $\tilde{p}_{S(i),S(j)}$ is approximated using comparisons between $S(i)$ and $S(j)$, by Hoeffding's inequality, w.p.$\geq 1-1/|S|^4$, $|\hat{\tilde{p}}_{S(i),S(j)} - \tilde{p}_{S(i),S(j)}| < \frac{3\varepsilon}{4}$, (since $S(i)$ and $S(j)$ are compared for $\frac{16}{\varepsilon^2}\log|S|^4$ times and $\tilde{p}_{S(i),S(j)}$ is approximated to the nearest multiple of ε). Since at most $|S|^2$ pairs are compared, by union bound w.p.$\geq 1-\frac{1}{|S|^2}$, all pair probabilities approximated using comparisons are correct to an accuracy of $3\varepsilon/4$.

Now we show that even when $\hat{\tilde{p}}_{S(k),S(i)}$ is given the same value as $\hat{\tilde{p}}_{S(k-1),S(i)}$ without using comparisons, $\tilde{p}_{S(k),S(i)}$ is approximated to an accuracy of $3\varepsilon/4+2\varepsilon'$.

Notice that for $S(1)$, when approximating pairwise probability $\tilde{p}_{S(k),S(1)}$ using comparisons if $\hat{\tilde{p}}_{S(k),S(1)} < \hat{\tilde{p}}_{S(k-1),S(1)}$, then $\hat{\tilde{p}}_{S(k),S(1)}$ is given same value as $\hat{\tilde{p}}_{S(k-1),S(1)}$. We first show that this process approximates $\tilde{p}_{S(k),S(1)}$ to an accuracy of $3\varepsilon/4+\varepsilon'$. Consider largest $l<k$ s.t. $\hat{\tilde{p}}_{S(l),S(1)} = \hat{\tilde{p}}_{S(k),S(1)}$. Notice that $\tilde{p}_{S(l),S(1)}$ is approximated using comparisons (because $\hat{\tilde{p}}_{S(l),S(1)} \neq \hat{\tilde{p}}_{S(l-1),S(1)}$) and hence $|\hat{\tilde{p}}_{S(l),S(1)} - \tilde{p}_{S(l),S(1)}| \leq 3\varepsilon/4$. Since by Lemma 72, $\tilde{p}_{S(k),S(1)} \geq \tilde{p}_{S(l),S(1)} - \varepsilon' \ \forall l<k$, it follows that

$$\begin{aligned}\hat{\tilde{p}}_{S(k),S(1)} &= \hat{\tilde{p}}_{S(l),S(1)} \\ &\leq \tilde{p}_{S(l),S(1)} + 3\varepsilon/4 \\ &\leq \tilde{p}_{S(k),S(1)} + 3\varepsilon/4 + \varepsilon'.\end{aligned}$$

Notice that since initially $\hat{\tilde{p}}_{S(k),S(1)}$ was assigned a value smaller than $\hat{\tilde{p}}_{S(k-1),S(1)}$ using comparisons, $\tilde{p}_{S(k),S(1)} < \hat{\tilde{p}}_{S(k-1),S(1)} + 3\varepsilon/4$ (since approximation using comparisons is correct to $3\varepsilon/4$). Therefore $\tilde{p}_{S(k),S(1)} < \hat{\tilde{p}}_{S(k),S(1)} + 3\varepsilon/4$. Hence $|\hat{\tilde{p}}_{S(k),S(1)} - \tilde{p}_{S(k),S(1)}| \leq 3\varepsilon/4+\varepsilon'$.

Therefore if comparisons are invoked (even when approximated probability is not due to comparisons), pairwise probability is approximated to an accuracy of $3\varepsilon/4+\varepsilon'$.

Now we consider the pairs for which no comparisons are invoked and show that even those pairwise probabilities are approximated to an accuracy of $3\varepsilon/4+2\varepsilon'$.

Suppose $\hat{\tilde{p}}_{S(k),S(i)}$ is copied same as $\hat{\tilde{p}}_{S(k-1),S(i)}$. Let j be the largest number $<k$ such that

$\hat{\tilde{p}}_{S(j),S(i)} \neq \hat{\tilde{p}}_{S(j-1),S(i)}$. Notice that $\hat{\tilde{p}}_{S(j),S(i)} = \hat{\tilde{p}}_{S(k),S(i)}$ and comparisons are used between $S(j)$ and $S(i)$ (since $\hat{\tilde{p}}_{S(j),S(i)} \neq \hat{\tilde{p}}_{S(j-1),S(i)}$). Therefore $|\tilde{p}_{S(j),S(i)} - \hat{\tilde{p}}_{S(j),S(i)}| \leq 3\varepsilon/4 + \varepsilon'$. Further by Lemma 72, $\tilde{p}_{S(j),S(i)} \leq \tilde{p}_{S(k),S(i)} + \varepsilon'$ and hence it follows that

$$\begin{aligned}
\hat{\tilde{p}}_{S(k),S(i)} &= \hat{\tilde{p}}_{S(j),S(i)} \\
&\leq \tilde{p}_{S(j),S(i)} + 3\varepsilon/4 + \varepsilon' \\
&\leq \tilde{p}_{S(k),S(j)} + 3\varepsilon/4 + 2\varepsilon'.
\end{aligned}$$

Similarly, it can be shown that $\hat{\tilde{p}}_{S(k),S(i)} \geq \tilde{p}_{S(k),S(j)} - 3\varepsilon/4 - 2\varepsilon'$. Therefore $|\hat{\tilde{p}}_{S(k),S(i)} - \tilde{p}_{S(k),S(i)}| \leq 3\varepsilon/4 + 2\varepsilon' = \varepsilon$.

Hence w.p.$\geq 1 - \frac{1}{|S|^2}$, all pairwise probabilities are approximated to an accuracy of ε.

We now bound the number of comparisons.

We first show that $0 \leq \hat{\tilde{p}}_{S(k),S(l)} \leq \hat{\tilde{p}}_{S(k+1),S(l)} \leq \hat{\tilde{p}}_{S(k+1),S(l-1)}$ $\forall k \geq l$. We prove this statement using induction. Notice that it is true for $l = 1$. We assume that it is true for l and prove correctness for $l+1$.

Notice that $0 = \hat{\tilde{p}}_{S(l+1),S(l+1)} \leq \hat{\tilde{p}}_{S(l+2),S(l)}$. Now we further assume that $0 \leq \hat{\tilde{p}}_{S(k),S(l+1)} \leq \hat{\tilde{p}}_{S(k+1),S(l)}$ and show that $0 \leq \hat{\tilde{p}}_{S(k+1),S(l+1)} \leq \hat{\tilde{p}}_{S(k+2),S(l)}$. Notice that if $\hat{\tilde{p}}_{S(k),S(l+1)} = \hat{\tilde{p}}_{S(k+1),S(l)}$, $\hat{\tilde{p}}_{S(k+1),S(l+1)} = \hat{\tilde{p}}_{S(k+1),S(l)} \leq \hat{\tilde{p}}_{S(k+2),S(l)}$. And if $\hat{\tilde{p}}_{S(k),S(l+1)} < \hat{\tilde{p}}_{S(k+1),S(l)}$, then $\tilde{p}_{S(k+1),S(l+1)}$ is approximated using comparisons and hence approximation accurate to $3\varepsilon/4$. Notice that $\hat{\tilde{p}}_{S(k+1),S(l+1)}$ can't be more than $\hat{\tilde{p}}_{S(k+1),S(l)}$ since if it happens, $\hat{\tilde{p}}_{S(k+1),S(l+1)} \geq \hat{\tilde{p}}_{S(k+1),S(l)} + \varepsilon \geq \hat{\tilde{p}}_{S(k),S(l+1)} + 2\varepsilon$ but $\hat{\tilde{p}}_{S(k+1),S(l+1)} \leq \tilde{p}_{S(k+1),S(l+1)} + 3\varepsilon/4 \leq \tilde{p}_{S(k),S(l+1)} + 3\varepsilon/4 + \varepsilon' \leq \hat{\tilde{p}}_{S(k),S(l+1)} + 3\varepsilon/2 + 3\varepsilon' < \hat{\tilde{p}}_{S(k),S(l+1)} + 2\varepsilon$ (contradiction). So $\hat{\tilde{p}}_{S(k+1),S(l+1)} \leq \hat{\tilde{p}}_{S(k+1),S(l)} \leq \hat{\tilde{p}}_{S(k+2),S(l)}$. Similarly it can be shown that $\hat{\tilde{p}}_{S(k+1),S(l+1)} \leq \hat{\tilde{p}}_{S(k),S(l+1)}$.

Consider the pairs $(\{S(1),S(i)\},\{S(2),S(i-1)\},\{(S(3),S(i-2)\},...,\{S(i/2),S(i/2+1)\})$. Since $1/2 \geq \hat{\tilde{p}}_{S(i),S(1)} \geq \hat{\tilde{p}}_{S(i-1),S(2)} \geq \hat{\tilde{p}}_{S(i-2),S(3)} \cdots \geq \hat{\tilde{p}}_{S(i/2+1),S(i/2)} \geq 0$ and all values are multiples of ε, $\hat{\tilde{p}}_{S(k),S(j)} \neq \hat{\tilde{p}}_{S(k-1),S(j+1)}$ only for $O(min(|S|, 1/\varepsilon))$ pairs $\{k, j\}$ such that

$k+j=i+1$. Hence only for $O(\min(|S|,1/\varepsilon))$ pairs $\{S(l),S(m)\}$ s.t. $l+m=i+2$, comparisons are used to approximate pairwise probabilities. This statement is true for all sum of indices and hence only for $O(|S|min(|S|,1/\varepsilon))$ pairs, comparisons are used and hence total comparisons used is $O\Big(\frac{|S|\min(|S|,1/\varepsilon)}{\varepsilon^2}\log|S|\Big)$. □

4.G Other Relevant Models

Researchers also considered models other than WST. For these models, one can still define maximum and ranking based on Borda scores, Copeland scores, or Von Neumann winner.

One such interesting model is considered in [HFCB08]: there is a probability distribution P over set $\mathcal{R}$ of all possible rankings and pairwise preferences can be defined based on this probability distribution,

$$p_{i,j}=\sum_{r\in\mathcal{R}}P(r)1_{i\succ j \text{ in } r}.$$

This model does not necessarily satisfy WST, however [HFCB08] shows that this model still satisfies some relation among pairwise probabilities

$$p_{i,j}\geq p_{i,k}+p_{k,j}-1. \tag{4.5}$$

It is natural to ask if a combination of WST and (4.5) guarantee a sub-quadratic complexity maxing algorithm. We give a negative answer to this question.

Recall that the model used in Section 4.3 to show lower bound of $\Omega(n^2)$ for WST is $p_{i,i+1}=1$ and $p_{i,j}\approx 0.5$ for $|i-j|\neq 1$. This model satisfies WST but not Equation (4.5). Yet we can slightly change the model to satisfy this new relation and derive the lower bound of $\Omega(n^2)$. The new model that results in lower bound of $\Omega(n^2)$ under WST and Equation (4.5) is $p_{i,i+1}=0.75$ and $p_{i,j}=0.5$ for $|i-j|\neq 1$. It is easy to check that this new model requires more comparisons than the model in Section 4.3 to find $\frac{1}{8}$-maximum. Hence lower bound of $\Omega(n^2)$ also

holds under the combination of WST and Equation 4.5.

Chapter 5

One Interview is Enough!: Optimal Sequential and Competitive

5.1 Introduction

Maximum selection (maximization) under probabilistic queries arises in numerous applications ranging from medical trials [Rob52] to social choice [CN91], to wireless channel band selection [AB10]. It has therefore been studied under various settings, including multi-armed [EDMM02, EDMM06, BMS09, KKS13, GGL12] and dueling bandits [SBFPH15, YBKJ12, FJO$^+$18]. These setups typically assume access to all alternatives at all times, so that each can be queried at will. They show that finding an ε-approximation of the maximum of n items with probability $\geq 1-\delta$ requires $\Theta\left(\frac{n}{\varepsilon^2}\log\frac{1}{\delta}\right)$ queries.

But what if the alternatives are presented in a streaming fashion? Is it still possible to find the best alternative? Can it be done using the same orderwise number of queries as in the non-streaming model? And what if the number of alternatives is not even known in advance as when data keeps streaming? Does this lack of knowledge increase the number of queries?

To better understand and further motivate the problem, recall the popular *Secretary*

problem, *e.g.,* [GM06]. A known number n of alternatives are presented in a uniformly random sequence, and after each alternative is presented, it is either accepted, ending the process, or rejected and the next alternative is presented. The goal is to maximize the probability that the best alternative is accepted.

This formulation assumes that n is known in advance, and that all the alternatives presented so far can be perfectly ranked. Both assumptions often fail to hold. Candidates can rarely be precisely ranked, and often n is not known in advance as additional candidates may join the pool.

We consider a more practical interview process that we call the *interview problem*, where the evaluation has an element of randomness, and an administrator would like to hire a near-best secretary with high probability, and to do so even when the number of candidates n is not known in advance. We consider two versions of the problem, one corresponding to traditional bandits and one to dueling bandits.

5.1.1 Traditional bandits

Here the administrator can interview one candidate at a time. We assume that each candidate interview consists of a sequence of queries, with each query providing probabilistic evidence about the candidate's merits. Each candidate i has a parameter $v_i \in [0,1]$ indicating the probability that the candidate answers each question correctly. To each question asked, candidate i gives a correct response with probability v_i, and distinct questions are answered independently. The confidence in the candidate's evaluation improves as the number of queries increases, yet when each candidate is interviewed, it is not clear how many queries would truly suffice. At each interview, the administrator can ask any number of queries to evaluate the current candidate but once the interview ends, the candidate can't be called for further evaluation. Adopting the conventional PAC formulation, for given $\varepsilon < 1/4$ and $\delta < 1/4$, we would like to find w.p. $\geq 1-\delta$ an ε-maximum candidate whose value is at most ε below that of the maximum value among all candidates. The goal is to minimize the total number of queries.

This is the traditional multi-armed bandits formulation, except that it is adapted for the streaming framework i.e., candidates come in a uniformly random sequence and one candidate can be interviewed at a time and once a candidate's interview is completed and sent away, he can't be recalled for further evaluation.

5.1.2 Dueling bandits

Here, in each interview the administrator can compare two candidates. To facilitate these "pairwise comparisons", the administrator is allowed to keep a "buffer" of one candidate and in each interview, he can compare the buffer candidate with a new candidate assigning them tasks to complete. For every independent task, candidate i will finish the task before candidate j with probability $p_{i,j}$, which is also referred to as the probability that i is *preferred* to j. If $p_{i,j} \geq \frac{1}{2}$, we say that i is *preferable* to j, denoted by $i \geq j$. Let $\tilde{p}_{i,j} = p_{i,j} - 1/2$ be the *centered preference probability*. Candidate i is ε-preferable to candidate j if $\tilde{p}_{i,j} \geq -\varepsilon$. Our goal here is: given $\varepsilon < 1/4$ and $\delta < 1/4$, to w.p.$\geq 1-\delta$, find an ε-*maximum* candidate that is ε-preferable to every other candidate. The confidence in the candidates' comparison improves as the number of tasks increases, yet during each interview, it is not clear how many tasks would truly suffice. After each interview the administrator decides whether the newly-compared candidate is eliminated and the "buffer" candidate continues, or the "buffer" candidate is eliminated and replaced by the new candidate. Once a candidate is eliminated, he can't be recalled. The process may stop at any time, and at that point, the "buffer" candidate is hired.

This is the dueling bandits formulation, except that it is adapted for the streaming framework i.e., candidates come in a uniformly random sequence and at a time only one interview can happen and after each interview a candidate is sent away and can never be recalled.

On the outset, this setting might look easier than the regular bandits setting since we can compare the current candidate with the "buffer" candidate, thereby getting more information about a previous ("buffer") candidate. But observe that this model is more general in the sense

that it has $\Theta(n^2)$ parameters whereas the traditional bandits setting has only n parameters.

Under this dueling bandits setting, to allow for the feasibility of existence of maximum, one needs to assume certain transitivity property among the elements. We assume one such property which has been used previously [FOPS17, FHO$^+$17].

The model is said to satisfy *Strong Stochastic Transitivity (SST)* if for any $i \geq j$ and $j \geq k$, $\tilde{p}_{i,k} \geq \max(\tilde{p}_{i,j}, \tilde{p}_{j,k})$.

Previous works also considered another property *Stochastic Triangle Inequality (STI)*. The model is said to satisfy STI, if for any $i \geq j$ and $j \geq k$, $\tilde{p}_{i,k} \leq \tilde{p}_{i,j} + \tilde{p}_{j,k}$.

Recall that under no constraints, maximization algorithms still require $\Theta\left(\frac{n}{\varepsilon^2}\log\frac{1}{\delta}\right)$ queries to find an ε-approximation of the maximum with probability $\geq 1-\delta$. We show that under both versions of the *interview problem*, $O\left(\frac{n}{\varepsilon^2}\log\frac{1}{\delta}\right)$ queries are sufficient to find an ε-best candidate with probability $\geq 1-\delta$. Therefore neither the streaming framework nor lack of a priori knowledge of n increases the orderwise query complexity. Hence we show that a candidate once sent away doesn't need to be called for further evaluation. One interview is enough!

5.1.3 Related Work

Researchers have considered severl extensions of the secretary problem, for example, minimizing the expected rank of the accepted alternative [Lin61] and finite memory [RS77]. For a detailed summary of extensions, we refer the reader to [Fre83].

Under traditional multi-armed bandit setting, [EDMM06, ZCL14] presented min-max optimal maximization algorithm that uses $O\left(\frac{n}{\varepsilon^2}\log\frac{1}{\delta}\right)$ queries and outputs ε approximation of maximum with probability $\geq 1-\delta$. Neither of these algorithms are streaming based or n-agnostic.

Under dueling bandits setting, for models with both SST and STI properties, [FOPS17] presented a min-max optimal maximization algorithm that uses $O\left(\frac{n}{\varepsilon^2}\log\frac{1}{\delta}\right)$ comparisons and outputs an ε-maximum with probability$\geq 1-\delta$. Their algorithm is neither streaming based nor n-agnostic. [FHO$^+$17] presented a min-max optimal maximization algorithm under a more

general model that satisfies only SST with comparison complexity of $O\left(\frac{n}{\varepsilon^2}\log\frac{1}{\delta}\right)$. This algorithm is neither streaming based nor n-agnostic. But in the process, for the same model [FHO$^+$17] also presented a sub optimal min-max maximization algorithm, SOFT-SEQ-ELIM with comparison complexity of $O\left(\frac{n}{\varepsilon^2}\log\frac{n}{\delta}\right)$. This algorithm is streaming based but not n-agnostic. [FJO$^+$18] presented min-max optimal maximization algorithms for more general models. [UCFN13, BFSH14, HSRW16] derived maximization algorithms for more general models under different maximum definitions.

Under a more restricted model, Plackett-Luce (PL) model, where each element i is associated with a value u_i and $p_{i,j} = \frac{u_i}{u_i+u_j}$, [SG19] derived a near instance optimal maximization algorithm that w.p.$\geq 1-\delta$, uses $O\left(\sum_{i=2}^{n}\frac{1}{\max(u_1-u_i,\varepsilon)^2}\log\frac{\log\frac{1}{\max(u_1-u_i,\varepsilon)}}{\delta}\right)$ comparisons and outputs an ε-maximum.

5.1.4 Questions

1) In the streaming model, we ask the following questions: a) What is the optimal query complexity under the traditional bandit settings? b) What is the optimal comparison complexity under the dueling bandit settings with SST? c) Will the answers change if n is not known in advance?

2) In the non-streaming model, we ask if we can extended near instance optimal results of [SG19] to more general models.

5.1.5 Results

1) In Theorems 75 and 88, for traditional bandits and dueling bandits models with SST respectively, we derive optimal n-agnostic streaming maximization algorithms that w.p.$\geq 1-\delta$ uses $O(\frac{n}{\varepsilon^2}\log\frac{1}{\delta})$ queries (or comparisons) and output an ε-maximum. Notice that since comparison complexity is orderwise same as that of lower bound for traditional multi-armed

bandits setting that need not be sequential and has a priori knowledge of n, we answered all questions in (1) above, with the same bound.

2) In Theorem 79, we show that the near instance optimal analysis of [SG19] can be extended to more general models with SST and STI properties with comparison complexity of $O\left(\sum_{i=2}^{n} \frac{1}{\max(\tilde{p}_{1,i},\varepsilon)^2} \log \frac{\log \frac{1}{\max(\tilde{p}_{1,i},\varepsilon)}}{\delta}\right)$.

5.1.6 Outline

In Section 5.2, we present TRAD-OPT-AGNOSTIC-SEQ, a min-max optimal streaming and n-agnostic maximization algorithm. In Section 5.3, we present OPT-AGNOSTIC-SEQ, a min-max optimal streaming and n-agnostic maximization algorithm. In Section 5.4, we present COMPETITIVE-MAX, a competitive maximization algorithm for models with SST and STI properties. In Section 5.5 we compare empirical performance of maximization algorithms. Finally, we provide our concluding remarks in Section 5.6.

5.2 Traditional Bandits Maximization

Before we present our min-max n-agnostic sequential maximization algorithm, we first present a simple n-agnostic sequential maximization algorithm SIMPLE-AGNOSTIC-SEQ with sub optimal query complexity and later build on it to present our min-max optimal maximization algorithm.

5.2.1 Simple Agnostic Sequential Maximization

Notice that if we can approximate all candidates' values to an additive accuracy of $\varepsilon/2$, then candidate with the highest approximated value will indeed be an ε-maximum. Using $\frac{2}{\varepsilon^2}\log\frac{1}{\delta'}$ queries, one can approximate a candidate's value to an additive accuracy of $\varepsilon/2$. To invoke union bound and ensure that w.p.$\geq 1-\delta$, all candidate values are approximated to an additive

accuracy of $\varepsilon/2$, one can use $\delta_i = 1/(2i^2)$ when evaluating the ith candidate. This results in an n-agnostic sequential maximization algorithm SIMPLE-AGNOSTIC-SEQ with query complexity of $O\left(\frac{n}{\varepsilon^2}\log\frac{n}{\delta}\right)$.

We present the pseudocode for SIMPLE-AGNOSTIC-SEQ in Appendix 5.A. In Lemma 73, we bound the query complexity and prove the correctness of SIMPLE-AGNOSTIC-SEQ.

Lemma 73. SIMPLE-AGNOSTIC-SEQ(S,ε,δ) *uses* $O\left(\frac{n}{\varepsilon^2}\log\frac{n}{\delta}\right)$ *queries and w.p.*$\geq 1-\delta$*, outputs an* ε*-maximum.*

Notice that SIMPLE-AGNOSTIC-SEQ is min-max optimal when $\delta < 1/n$ but incurs an extra $\log n$ multiplicative factor for constant δ. We now explore whether that extra $\log n$ factor is necessary.

5.2.2 Optimal Agnostic Sequential Maximization

We first identify sufficient conditions for correctness of any sequential maximization algorithm and point out shortcomings of SIMPLE-AGNOSTIC-SEQ. We then propose a min-max optimal algorithm reducing the overhead of SIMPLE-AGNOSTIC-SEQ.

Lets assume that a maximization algorithm maintains r, a proxy for the output if sequence of candidates ends. Notice that if then the output will be an ε-maximum if one can ensure:

1. the original underlying value v_r of r never decreases,
2. when the best candidate is being interviewed, if value of the best candidate is at least ε more than that of the r, then r is updated to the best candidate.

OPT-APPROX

To make sure that value of r never gets worse, one can approximate all candidate values within $\pm\varepsilon/4$ and update r only if current candidate's approximated value is $\varepsilon/2$ more than that

of r. Notice that if one wants to approximate all candidates' values to $\pm\varepsilon/4$, then one cannot escape the extra $\log n$ factor. Another way to ensure that r never gets worse is by never updating r but then the second sufficient property that r should be updated if current candidate's value is at least ε more than that of r will be violated. One way to circumvent this issue is to approximate current candidate's value to $\pm\varepsilon/4$ with probability $1-\Theta(\delta/i^2)$ only if r is going to be updated. Notice that SIMPLE-AGNOSTIC-SEQ approximates the ith candidate's value with probability $1-\delta/(2i^2)$, hence uses too many queries in one shot. We instead divide the queries into rounds, decreasing confidence paramter (thereby increasing confidence in evaluation) with each round such that total queries is still of the same order. In a given interview, we move to the next round of queries only if in all previous rounds, candidate's approximated value is at least $\varepsilon/2$ more than that of r's approximated value. This helps in terminating the interview much earlier than asking all questions in one shot. In the last round, confidence parameter δ' is such that the value is approximated to $\pm\varepsilon/4$ w.p.$\geq 1-O(\delta/\imath^2)$. We present OPT-APPROX that approximates the value of a candidate c using rounds of queries.

Algorithm 35 OPT-APPROX

inputs
  candidate c, bias ε, confidence δ, candidate number i, max value m_v
$v_c \leftarrow 0$
for $t = 0$ to $\max(1, \lceil \log\log_{\frac{1}{\delta}} i^2 \rceil)$ **do**
  $\delta_t = \frac{\delta^{2^t+1}}{8}$
  Ask c $\frac{8}{\varepsilon^2}\log\frac{1}{\delta_t}$ queries. $\hat{v}_c \leftarrow$ Fraction of correct response
  if $\hat{v}_c < m_v + \varepsilon/2$ **then**
    break
  end if
end for
return $\hat{v}_c$

In the Lemma below, we provide guarantees for OPT-APPROX.

Lemma 74. OPT-APPROX$(c,\varepsilon,\delta,i,m_v)$ *uses* $O\left(\frac{1}{\varepsilon^2}\log\frac{i}{\delta}\right)$ *queries and outputs* $\hat{v}_c$ *s.t. if* $v_c < m_v+\varepsilon/4$, *then w.p.*$\geq 1-\delta/(8i^2)$, $\hat{v}_c < m_v+\varepsilon/2$ *and if* $v_c \geq m_v+3\varepsilon/4$, *then w.p.*$\geq 1-\delta/4$,

$\hat{v}_c \geq m_v + \varepsilon/2$.

TRAD-OPT-AGNOSTIC-SEQ

Now we present our main algorithm TRAD-OPT-AGNOSTIC-SEQ that uses OPT-APPROX to approximate value v_c of candidate c with $\hat{v}_c$ and updates the anchor if $\hat{v}_c \geq m_v + \varepsilon/2$.

Algorithm 36 TRAD-OPT-AGNOSTIC-SEQ

inputs
  Set S, bias ε, confidence δ
Initialize:max value $m_v = -\infty$, anchor $r \leftarrow S(1)$, candidate number $i \leftarrow 0$
while $S \neq \emptyset$ **do**
  $c \leftarrow S(1)$, $S = S \setminus \{c\}$, $i \leftarrow i+1$
  $\hat{v}_c \leftarrow$ OPT-APPROX$(c, \varepsilon, \delta, i, m_v)$
  if $\hat{v}_c \geq$ max-value $+ \varepsilon/2$ **then**
    max-value $\leftarrow \hat{v}_c$, $r \leftarrow c$
  end if
end while
return r

Lemma 74 implies the properties 1 and 2 which in turn guarantees that the output is an ε-maximum.

Using randomness in sequence of candidates, we show that the best candidate of all the seen candidates changes rarely as we move along the sequence. This helps to show that OPT-APPROX rarely reaches higher rounds and hence we obtain a bound on queries used.

In Theorem 75, we provide guarantees for TRAD-OPT-AGNOSTIC-SEQ.

Theorem 75. *W.p.* $\geq 1-\delta$, TRAD-OPT-AGNOSTIC-SEQ(S, ε, δ) *uses* $O\left(\frac{n}{\varepsilon^2}\log\frac{1}{\delta}\right)$ *queries and outputs an* ε*-maximum.*

5.3 Dueling Bandits Sequential Maximization

5.3.1 Tools

We use subroutine COMPARE2 [FHO$^+$17] as a building block in our maximization algorithms. For the reader's convenience, we provide a brief outline of COMPARE2 here and state its guarantees in Lemma 76. We also present the algorithm COMPARE2 in Appendix 5.B.

For $\varepsilon_u > \varepsilon_l$, COMPARE2$(i, j, \varepsilon_l, \varepsilon_u, \delta)$ compares elements i and j for $O\Big(\frac{1}{(\varepsilon_u-\varepsilon_l)^2}\log\frac{1}{\delta}\Big)$ times and deems if $\tilde{p}_{i,j} \le \varepsilon_l$ (returns 1) or $\tilde{p}_{i,j} \ge \varepsilon_u$ (returns 2). The guarantees are presented in Lemma 76.

Lemma 76 (Lemma 1 [FHO$^+$17])**.** *For $\varepsilon_u > \varepsilon_l$,* COMPARE2$(i, j, \varepsilon_l, \varepsilon_u, \delta)$ *uses $\le \frac{2}{(\varepsilon_u-\varepsilon_l)^2}\log\frac{2}{\delta}$ comparisons and if $\tilde{p}_{i,j} \le \varepsilon_l$, then w.p.$\ge 1-\delta$, returns* 1*, else if $\tilde{p}_{i,j} \ge \varepsilon_u$, w.p.$\ge 1-\delta$, returns* 2*.*

5.3.2 Agnostic Version of SOFT-SEQ-ELIM [FHO$^+$17]

Recall that under models with SST property, SOFT-SEQ-ELIM is a sub optimal maximization algorithm and is sequential and requires the knowledge of n a priori.

We first describe an outline of SOFT-SEQ-ELIM and present an easy fix to make it n-agnostic with orderwise same sample complexity. SOFT-SEQ-ELIM starts with the first element as the anchor r, sequentially compares r with elements of S using COMPARE2$(S(i), r, 0, \varepsilon, \delta/n)$, and updates r with $S(i)$ if COMPARE2 returns 2. This ensures that with probability $1-\delta/n$: 1) the updated anchor is at least as good as the previous anchor, and 2) the updated anchor is ε-preferable to $S(i)$. These two key properties along with SST property and the union bound, ensure that w.p.$\ge 1-\delta$, the final anchor is an ε-maximum. Notice that to ensure that the total error probability is bounded by δ, SOFT-SEQ-ELIM uses each instance of COMPARE2 with confidence parameter δ/n and hence requires knowing n beforehand. A simple fix is to use confidence parameter $\delta/(2i^2)$ (observe that $\sum_{i=1}^{\infty}\delta/(2i^2) \le \delta$) when using ith instance of COMPARE2 and hence does

not require knowing the value of n. Now we present maximization algorithm AGNOSTIC-SEQ with this fix applied to SOFT-SEQ-ELIM. Notice that even nth instance of COMPARE2 uses $O\left(\frac{1}{\varepsilon^2}\log\frac{n}{\delta}\right)$ comparisons and hence AGNOSTIC-SEQ has orderwise same comparison complexity as SOFT-SEQ-ELIM. The pseudocode for AGNOSTIC-SEQ is provided in Appendix 5.C.

In the Lemma 77, we prove the correctness and bound the comparison complexity of AGNOSTIC-SEQ.

Lemma 77. *Under SST model,* AGNOSTIC-SEQ(S,ε,δ) *uses* $O\left(\frac{n}{\varepsilon^2}\log\frac{n}{\delta}\right)$ *comparisons and w.p.*$\geq 1-\delta$*, outputs an* ε*-maximum.*

Observe that AGNOSTIC-SEQ is n-agnostic and min-max optimal for $\delta \leq \frac{1}{n}$ but requires an extra multiplicative factor of $\log n$ comparisons than the known lower bound for constant δ.

5.3.3 Optimal Agnostic Sequential Maximization

In this subsection, for models with SST property, we present maximization algorithm OPT-AGNOSTIC-SEQ that is both sequential and n-agnostic and yet uses orderwise same comparisons as the min-max optimal maximization algorithm that has the knowledge of n and is not necessarily sequential. Hence OPT-AGNOSTIC-SEQ is also a min-max optimal maximization algorithm.

Due to lack of space, here we only provide a brief outline of our algorithm and state the main result. The motivation and analysis is very similar to that of TRAD-OPT-AGNOSTIC-SEQ and is presented in deltail in Appendix 5.F.

OPT-ANCHOR-UPDATE

Observe that in each instance to update the anchor, AGNOSTIC-SEQ uses COMPARE2 with confidence parameter $\delta/(8i^2)$. Here we present an alternative OPT-ANCHOR-UPDATE for using COMPARE2 in one shot. Similar to OPT-APPROX, within each instance of OPT-ANCHOR-UPDATE, we use multiple rounds of COMPARE2, decreasing the confidence parameter with each

consecutive round such that overall comparisons used over all rounds are orderwise same as comparisons used in a single instance of COMPARE2 with confidence parameter $\Theta(\frac{\delta}{i^2})$. Within each instance, we move to the next COMPARE2 round only if the previous round returna 2. This helps in terminating much earlier than if only one round of COMPARE2 is used.

Algorithm 37 OPT-ANCHOR-UPDATE

inputs
  element e, element f, bias ε, confidence δ, number i
Initialize: $t \leftarrow 0$, $a \leftarrow 2$
while $a = 2$ and $t < \max(2, \log\log_{\frac{1}{\delta}} i^2 + 1)$ **do**
  $a \leftarrow$ COMPARE2$(e, f, 0, \varepsilon, \delta^{2^t+1}/8)$
  $t \leftarrow t + 1$
end while
if $a = 1$ **then**
  return f
else
  return e
end if

OPT-AGNOSTIC-SEQ

We now present our main algorithm OPT-AGNOSTIC-SEQ that uses OPT-ANCHOR-UPDATE as subroutine to update the anchor.

Algorithm 38 OPT-AGNOSTIC-SEQ

inputs
  Set S, bias ε, confidence δ
anchor $r \leftarrow S(1)$, $S = S \setminus \{r\}$, candidate number $i \leftarrow 0$
while $S \neq \emptyset$ **do**
  $c \leftarrow S(1)$, $S = S \setminus \{c\}$, $i \leftarrow i + 1$
  $r \leftarrow$ OPT-ANCHOR-UPDATE$(c, r, \varepsilon, \delta, i)$
end while
return r

In the below Theorem, we provide guarantees for OPT-AGNOSTIC-SEQ.

Theorem 78. *Under SST models, w.p.* $\geq 1-\delta$, OPT-AGNOSTIC-SEQ (S,ε,δ) *uses* $O\left(\frac{n}{\varepsilon^2}\log\frac{1}{\delta}\right)$ *comparisons and outputs an* $\varepsilon-$*maximum.*

5.4 Competitive Maximization

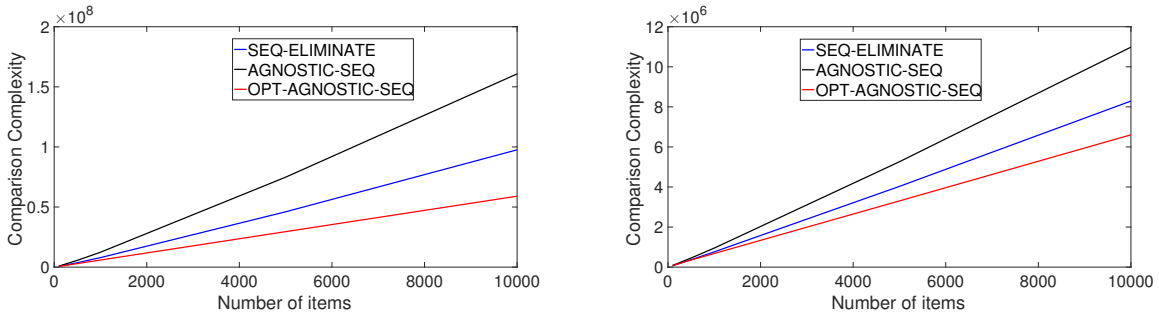
We now consider non-sequential dueling bandits setting and present a competitive maximization algorithm. We consider the model that satisfies both SST and STI conditions. Under this model we use an algorithm similar to PAC-WRAPPER [SG19] and show that this algorithm is competitive even under model more general than PL model considered in [SG19].

The algorithm works as follows. Consider a min-max optimal algorithm PAC-MAX that for given ε and δ parameters, w.p.$\geq 1-\delta$, uses $O(\frac{n}{\varepsilon^2}\log\frac{1}{\delta})$ comparisons and outputs an ε-maximum. Notice that there are several min-max optimal algorithms OPT-AGNOSTIC-SEQ and [FOPS17, FHO$^+$17, FJO$^+$18] for models with both SST and STI constraints. COMPETITIVE-MAX sequentially finds $1/2^{i+1}$-maximum of the remaining elements and uses this element to prune the elements that are not $1/2^i$-maximum. Let r_i be an $1/2^{i+1}$-maximum and e be such that it is not $1/2^i$-maximum. Observe that by SST and STI, e is not $1/2^{i+1}$-preferable to r_i. Hence w.p.$\geq 1-\delta_i$, COMPARE2$(e,f,0,1/2^{i+1},\delta_i)$ will return 2 and hence e will be eliminated. Hence each such bad element will be eliminated with probability δ_i. Every element e s.t., $\tilde{p}_{t_1,e}\approx 1/2^t$ will be eliminated in approximately t rounds and hence in future rounds the comparisons won't be used on e providing us the competitive upper bound.

Theorem 79. *W.p.* $\geq 1-\delta$, COMPETITIVE-MAX(S,ε,δ) *uses* $O\left(\sum_{i=2}^{n}\frac{1}{\max(\tilde{p}_{1,i},\varepsilon)^2}\log\frac{\log\left(\frac{1}{\max(\tilde{p}_{1,i},\varepsilon)}\right)}{\delta}\right)$ *(where* 1 *is assumed to be the best) comparisons and outputs an* ε*-maximum.*

Algorithm 39 COMPETITIVE-MAX

inputs
  Set S, bias ε, confidence δ
Initialize: $i = 1$
while $|S| > 1$ and $i < \log_2 \frac{1}{\varepsilon}$ **do**
  $\varepsilon_i = 1/2^{i+1}$, $\delta_i = \delta/(60i^3)$
  $r_i \leftarrow$ PAC-MAX$(S, \varepsilon_i, \delta_i)$
  Initialize: $Q = \emptyset$
  for e in S **do**
    if COMPARE2$(r_i, e, 0, \varepsilon_i, \delta_i) == 2$ **then**
      $Q = Q \bigcup \{e\}$
    end if
  end for
  $S = S \setminus Q$
end while
return PAC-MAX$(S, \varepsilon, \delta/30)$



(a) When all elements are equal

(b) When there is a strict ordering

Figure 5.1: Comparison of Maximization Algorithms

5.5 Experiments

In this section, we compare the performance of various sequential maximization algorithms SOFT-SEQ-ELIM [FHO+17], AGNOSTIC-SEQ and OPT-AGNOSTIC-SEQ. Note that SOFT-SEQ-ELIM uses the knowledge of n whereas AGNOSTIC-SEQ and OPT-AGNOSTIC-SEQ are n-agnostic. Further recall that SOFT-SEQ-ELIM and AGNOSTIC-SEQ are sub-optimal with query complexity of $O\left(\frac{n}{\varepsilon^2}\log\frac{n}{\delta}\right)$ and OPT-AGNOSTIC-SEQ is optimal with query complexity of $O\left(\frac{n}{\varepsilon^2}\log\frac{1}{\delta}\right)$. Experiments in [FHO+17, FJO+18] demonstrate that SOFT-SEQ-ELIM performs better than other maximization algorithms. Hence we don't compare with other maximization

algorithms. In all the experiments in this section, we try to find an 0.05-maximum with $\delta = 0.1$. All results are averaged over 100 runs.

We first consider the model where all items are essentially equal i.e., $p_{i,j} = 1/2 \ \forall i, j$. Figure 5.1(a) show the performance of sequential maximization algorithms for this model. Notice that OPT-AGNOSTIC-SEQ uses significantly less comparisons than both SOFT-SEQ-ELIM and AGNOSTIC-SEQ. Notice that since AGNOSTIC-SEQ is an agnostic version of SOFT-SEQ-ELIM, AGNOSTIC-SEQ uses more comparisons than SOFT-SEQ-ELIM.

We now consider the model where $p_{i,j} = 0.6 \ \forall i < j$ same as in [YJ11, FOPS17, FHO$^+$17, FJO$^+$18]. Figure 5.1(b) presents the performance of sequential maximization algorithms for this model. Notice again that OPT-AGNOSTIC-SEQ uses less comparisons than SOFT-SEQ-ELIM, that in turn uses fewer comparisons than AGNOSTIC-SEQ.

Since [FHO$^+$17, FJO$^+$18] showed that SOFT-SEQ-ELIM outperforms other maximization algorithms and empirical performance of OPT-AGNOSTIC-SEQ is better than SOFT-SEQ-ELIM, OPT-AGNOSTIC-SEQ outperforms even non-sequential maximization algorithms.

5.6 Conclusion and Future Work

We presented the first optimal sequential probabilistic maximization algorithm that works even without a-priori knowledge of number of items. The algorithm has linear complexity both under traditional- and dueling (with SST property)- bandits frameworks. We also present a modification of the maximization algorithm that is *competitive* under dueling bandits setting with SST and STI properties. In the future, we propose to extend these works to more general settings.

5.7 Acknowledgement

Chapter 5, in full, has been submitted for the publication of the material as it may appear in *Advances in Neural Information Processing Systems*. Falahatgar, Moein, Alon Orlitsky, and Venkatadheeraj Pichapati, 2019. The dissertation author was the primary investigator and author of this paper.

5.A Algorithm SIMPLE-AGNOSTIC-SEQ

SIMPLE-AGNOSTIC-SEQ maintains anchor r, a proxy for the candidate with highest approximated score so far. SIMPLE-AGNOSTIC-SEQ updates r with current candidate if their approximated score is more than that of r. After interviewing the final candidate SIMPLE-AGNOSTIC-SEQ outputs r

Algorithm 40 SIMPLE-AGNOSTIC-SEQ

inputs
  Set S, bias ε, confidence δ
Initialize:max value $m_v = -\infty$, anchor $r \leftarrow S(1)$, $i \leftarrow 0$
while $S \neq \emptyset$ **do**
  $c \leftarrow S(1)$, $S = S \setminus \{c\}$, $i \leftarrow i+1$
  Ask c, $\frac{2}{\varepsilon^2}\log\frac{4i^2}{\delta}$ queries. $\hat{v}_c \leftarrow$ Fraction of correct response
  if $\hat{v}_c > m_v$ **then**
    $m_v \leftarrow \hat{v}_c$, $r \leftarrow c$
  end if
end while
return r

5.B Algorithm COMPARE2

Algorithm 41 COMPARE2

inputs

element i, element j, bias lower limit $\varepsilon_l \geq 0$, bias upper limit $\varepsilon_u > \varepsilon_l$, confidence δ

initialize

$\varepsilon_m = (\varepsilon_l + \varepsilon_u)/2$, $\hat{\tilde{p}}_{i,j} \leftarrow 0$, $\hat{c} \leftarrow \frac{1}{2}$, $t \leftarrow 0$, $w \leftarrow 0$

while $|\hat{\tilde{p}}_{i,j} - \varepsilon_m| \leq \hat{c}$ and $t \leq \frac{2}{(\varepsilon_u - \varepsilon_l)^2} \log \frac{2}{\delta}$ **do**

Compare i and j

if i wins **then**

$w \leftarrow w + 1$

end if

$t \leftarrow t + 1$

$\hat{\tilde{p}}_{i,j} \leftarrow \frac{w}{t} - \frac{1}{2}$, $\hat{c} \leftarrow \sqrt{\frac{1}{2t} \log \frac{4t^2}{\delta}}$

end while

if $\hat{\tilde{p}}_{i,j} \leq \varepsilon_m$ **then**

return 1

end if

return 2

5.C Algorithm AGNOSTIC-SEQ

Algorithm 42 AGNOSTIC-SEQ

inputs
  Set S, bias ε, confidence δ
$r \leftarrow S(1)$, $S = S \setminus \{r\}$, $i \leftarrow 0$
while $S \neq \emptyset$ **do**
  $c \leftarrow S(1)$, $S = S \setminus \{c\}$, $i \leftarrow i+1$
  if COMPARE2$(c, r, 0, \varepsilon, \frac{\delta}{2i^2}) = 2$ **then**
    $r \leftarrow c$
  end if
end while
return r

5.D Proofs for Section 5.2

5.D.1 Proofs for Subsection 5.2.1

Proof of Lemma 73

Proof. Since we use $\frac{2}{\varepsilon^2}\log\frac{4i^2}{\delta}$ queries to approximate value of ith candidate, by Hoeffding's inequality, w.p.$\geq 1-\delta/(2i^2)$, ith candidate's value is approximated to $\pm\varepsilon/2$. By union bound, w.p.

$$\geq 1-\sum_{i=1}^{\infty}\delta/(2i)^2 \geq 1-\delta,$$

all candidates' values are approximated to $\pm\varepsilon/2$. Since SIMPLE-AGNOSTIC-SEQ approximates all candidates' values to $\pm\varepsilon/2$ and returns the candidate with highest approximated value, the output is an ε-maximum. □

5.D.2 Proofs for Subsection 5.2.2

Proof of Lemma 74

Proof. We first bound the number of comparisons. The comparisons used over all rounds is

$$\begin{aligned}
\sum_{i=1}^{\max(1,\lceil \log\log_{1/\delta} i^2\rceil)} O\left(\frac{1}{\varepsilon^2}\log\frac{8}{\delta^{2^t+1}}\right) &= O\left(\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\sum_{i=1}^{\max(1,\lceil \log\log_{1/\delta} i^2\rceil)}(2^t+1)\right)\\
&= O\left(\frac{2^{1+\max(1,\log\log_{1/\delta} i^2)}}{\varepsilon^2}\log\frac{1}{\delta}\right)\\
&= O\left(\frac{2\max(1,\log_{1/\delta} i^2)}{\varepsilon^2}\log\frac{1}{\delta}\right)\\
&= O\left(\frac{1}{\varepsilon^2}\left(\log\frac{1}{\delta}+\log\frac{1}{i^2}\right)\right)\\
&= O\left(\frac{1}{\varepsilon^2}\log\frac{i}{\delta}\right).
\end{aligned}$$

Now we show that if $v_c < m_v + \varepsilon/4$, then w.p.$\geq 1-\delta/(8i^2)$, $\hat{v}_c < m_v+\varepsilon/2$. Notice that output $v_c > m_v+\varepsilon/2$ only if it goes through last round $T = \max(1, \lceil \log\log_{\frac{1}{\delta}} i^2\rceil)$. Notice that

$$\delta_T = \frac{\delta^{2^T+1}}{8} \leq \frac{\delta^{2^{\max(1,\log\log_{\frac{1}{\delta}} i^2)}+1}}{8} \leq \frac{\delta}{8i^2}.$$

Hence in last round T, by Hoeffding's inequality, w.p.$\geq 1-\frac{\delta}{8i^2}$, $\hat{v}_c < v_c+\varepsilon/4 < m_v+\varepsilon/2$.

Now, we show that if $v_c \geq m_v+3\varepsilon/4$, then w.p.$\geq 1-\delta/4$, $\hat{v}_c \geq m_v+\varepsilon/2$. By Hoeffding's inequality, in round t, w.p. $\geq 1-\delta_t$,

$$\hat{v}_c > v_c - \varepsilon/4 \geq m_v+\varepsilon/2.$$

Hence by union bound w.p.

$$\geq 1 - \sum_{t=0}^{\infty} \delta_t \geq 1 - \sum_{t=0}^{\infty} \delta^{2^t+1}/8 \geq 1 - \delta/4$$

, in all rounds $\hat{v}_c > m_v + \varepsilon/2$ and hence final $\hat{v}_c > m_v + \varepsilon/2$. □

Proof of Theorem 75

Proof. We divide the proof into following steps:

1. W.p.$\geq 1 - \delta/2$, for all candidates if last round is invoked then the candidate's value is approximated to an accuracy of $\pm\varepsilon/4$.
2. Using 1, m_v is always within $\pm\varepsilon/4$ of anchor's value
3. Using 1 and 2, anchor's value never gets worse
4. Using 2 and Lemma 74, if candidate's value is at least ε more than that of anchor, then w.p.$\geq 1 - \delta/4$, anchor will be updated to chandidate
5. Using 3 and 4, output is an ε-maximum.
6. Using Lemma 74, TRAD-OPT-AGNOSTIC-SEQ uses $O\left(\frac{n}{\varepsilon^2}\log\frac{n}{\delta}\right)$ queries and hence optimal for $\delta \leq 200/n$.
7. Using 1, 3 and randomness of sequence, we lower bound m_v after l candidates
8. Using 7, we show that not many elements reach higher rounds.
9. Using 8, we bound comparisons for $\delta > 200/n$

Total probability of error comes from steps 1, 4 and 9 (which is union bound over 7 and 8). Step 1 incurs $\delta/2$ error probability, steps 4 and 9 incur $\delta/4$ error probability each.

We prove sequence of steps assuming previous steps hold.

We first prove correctness of TRAD-OPT-AGNOSTIC-SEQ.

Step 1 By Hoeffding's inequality, for ith candidate if last round $T = \max(1, \lceil \log\log_{\frac{1}{\delta}} i^2 \rceil)$ is invoked, then w.p.$\geq 1 - \delta_T \geq 1 - \frac{\delta}{4i^2}$, value of ith candidate is approximated to an accuracy of $\varepsilon/4$.

Hence by union bound, w.p.$\geq 1 - \sum_{i=1}^{\infty} \delta/(4i^2) \geq 1 - \delta/2$, for all candidates if last round is invoked, then candidate's value is approximated to an accuracy of $\pm\varepsilon/4$. From here, we assume that this event (we call it event E) has happened and account $\delta/2$ for probability error.

Step 2 Notice that anchor updates only when last round is invoked hence all anchors' values are approximated to an accuracy of $\pm\varepsilon/4$. Observe that if r is current anchor then $|m_v - v_r| \leq \varepsilon/4$.

Step 3 We now show that anchor's value only gets better.

Now if $v_c < v_r$, and if last round is invoked, $\hat{v}_c \leq v_c + \varepsilon/4 < v_r + \varepsilon/4 \leq m_v + \varepsilon/2$, hence c won't become new anchor.

Therefore anchor's value only gets better.

Step 4 By Lemma 74 if $v_c \geq v_r + \varepsilon \geq m_v + 3\varepsilon/4$, w.p.$\geq 1 - \delta/4$, $\hat{v}_c \geq m_v + \varepsilon/2$ and hence anchor will be updated to c.

Step 5 Now consider the best candidate b, after instance of b is done, w.p. $\geq 1 - \delta/4$, $v_r \geq v_b - \varepsilon$ and since anchor only gets better, the final output will be an ε-maximum.

Now we bound the number of comparisons.

Step 6 From Lemma 74, OPT-APPROX uses $O\left(\frac{1}{\varepsilon^2}\log\frac{i}{\delta}\right)$ queries for ith candidate and hence in total, $O\left(\frac{1}{\varepsilon^2}\log\frac{n}{\delta}\right)$ queries are used which is optimal for $\delta \leq 200/n$. From here, we assume that $\delta > 200/n$ and bound the comparisons.

Step 7 We use some definitions to complete the proof.

Definition 80. *Let the candidates be ranked according to their values with lowest valued candidate getting rank n and the highest valued candidate getting rank* 1 *(ties are broken arbitrarily). Let t_k denote the candidate with rank k. Further let $c_{k,\varepsilon,\delta,l,\alpha}$ denote maximum of all values u s.t. w.p. $\geq \alpha$,* OPT-APPROX$(t_k,\varepsilon,\delta,l,u)$ *outputs* $\geq u+\varepsilon/2$.

Notice that the probability that OPT-APPROX$(e,\varepsilon,\delta,l,u)$ *outputs* $\geq u+\varepsilon/2$ *is an increasing function with v_e and a decreasing function with u and hence for all $k' \leq k$ and $u' \leq c_{k,\varepsilon,\delta,l,\alpha}$, w.p. $\geq \alpha$,* OPT-APPROX$(t_{k'},\varepsilon,\delta,l,u')$ *outputs* $\geq u'+\varepsilon/2$. *Similarly, for all $k' \geq k$ and $u' \geq c_{k,\varepsilon,\delta,l,\alpha}$, the probability that* OPT-APPROX$(t_{k'},\varepsilon,\delta,l,u')$ *outputs* $\geq u'+\varepsilon/2$ *is* $\leq \alpha$.

Notice that $c_{k,\varepsilon,\delta,l,\alpha} \leq v_{t_k}-\varepsilon/4$ for all $\alpha > 0$ since in last round all values are approximated to an accuracy of $\pm\varepsilon/4$ and if $u > v_{t_k}-\varepsilon/4$ then w.p.$\geq \alpha > 0$, OPT-APPROX$(t_{k'},\varepsilon,\delta,l,u)$ needs to output $\geq u+\varepsilon/2 > v_{t_k}+\varepsilon/4$ which voilates the event E.

Now we show that for all l,k,n w.p.$\geq 1-2e^{-\sqrt{\frac{kl}{4n}}}$, after lth candidate in sequence,

$$m_v \geq c_{k,\varepsilon,\delta,l,\alpha}.$$

By Chernoff bound, w.p.$\geq 1-e^{\frac{kl}{4n}}$, in first l candidates there will be at least $\frac{kl}{4n}$ candidates from k top ranked (highest valued) candidates. Let $u = c_{k,\varepsilon,\delta,l,\sqrt{\frac{4n}{kl}}}$. Notice that during TRAD-OPT-AGNOSTIC-SEQ, m_v only increases and increases by at least $\varepsilon/2$ whenever anchor changes. If $m_v < u$ and the candidate is one from top k candidates, then w.p.$\geq \alpha$ the candidate will become new anchor and new $m_v \geq v_{t_k}-\varepsilon/4 \geq u$. Since there are at least $\frac{lk}{4n}$ such candidates in first l candidates the probability that m_v less than u not getting replaced with value greater than u is

$$\left(1-\frac{lk}{4n}\right)^{\sqrt{\frac{4n}{kl}}} \leq e^{-\sqrt{\frac{kl}{4n}}}.$$

Hence by union bound, w.p. $\geq 1-2e^{-\sqrt{\frac{kl}{4n}}}$, after lth candidate in sequence

$$m_v \geq c_{k,\varepsilon,\delta,l,\sqrt{\frac{4n}{kl}}}.$$

Step 8 Since m_v only increases, m_v for canidate $l' > l$ is more than $c_{k,\varepsilon,\delta,l,\sqrt{4n},kl}$. Hence probability that for $k' > k$ and $l' > l$, OPT-APPROX$(t_{k'},\varepsilon,\delta,l',m_v)$ reaches more than $\max(2,\lceil \log\log_{\frac{1}{\delta}} l^2 \rceil + 1)$ rounds is $\leq \sqrt{\frac{4n}{kl}}$. Hence by chernoff bound, after lth candidate, w.p.$\geq 1-e^{-n\sqrt{\frac{4n}{kl}}}$, out of all candidates ranked outside top k, the number of elements that reach more than $\max(2,\lceil \log\log_{\frac{1}{\delta}} l^2 \rceil + 1)$ rounds is $\leq n\sqrt{\frac{36n}{kl}}$.

Now, using union bound, w.p.$\geq 1-2e^{-\sqrt{\frac{kl}{4n}}}-e^{-n\sqrt{\frac{4n}{kl}}}$, over all instances of OPT-APPROX during TRAD-OPT-AGNOSTIC-SEQ(S,ε,δ), , the number of times the round $\max(2,\lceil \log\log_{\frac{1}{\delta}} l^2 \rceil + 1)$ reached is $\leq k+l+n\sqrt{\frac{36n}{kl}}$.

Step 9 Now we complete by selecting sequence of values

$$l_i = i^6$$
$$k_i = \frac{4n\left(\log\left(\frac{2^{i^2}}{\delta^2}\right)\right)^2}{l_i},$$

for $\log\frac{1}{\delta} \leq i \leq \log n$.

$$l_{\log n+1} = n.$$

W.p.$\geq 1-2e^{-\sqrt{\frac{k_i l_i}{4n}}}-e^{-n\sqrt{\frac{4n}{k_i l_i}}}$, number of elemets that are queired $\Omega\left(\frac{1}{\varepsilon^2}\log\frac{l_i}{\delta}\right)$ times is $O\left(l_i+k_i+n\sqrt{\frac{4n}{l_i k_i}}\right)$. Hence by union bound, $\geq 1-\sum_i(e^{1-\sqrt{\frac{l_i k_i}{4n}}}-e^{-n\sqrt{\frac{4n}{l_i k_i}}})$, and using upper sum,

the total number of queries is

$$\frac{1}{\varepsilon^2}O\left(n\log\frac{1}{\delta}+\sum_{i=\log\frac{1}{\delta}}^{\log n}\left(l_i+k_i+n\sqrt{\frac{n}{l_ik_i}}\right)\left(\log\frac{l_{i+1}}{\delta}\right)\right).$$

We first bound the probability of event,

$$\begin{aligned}\sum_{i=\log\frac{1}{\delta}}^{\log n}(2e^{-\sqrt{\frac{l_ik_i}{4n}}}+e^{-n\sqrt{\frac{4n}{l_ik_i}}}) &= \sum_{i=\log\frac{1}{\delta}}^{\log n}(2e^{-\log\frac{2^{i^2}}{\delta^2}}+e^{-n/(\log\frac{2^{i^2}}{\delta^2})}\\ &\leq \sum_{i=\log\frac{1}{\delta}}^{\log n} e\frac{\delta^2}{2^{i^2}}+e^{-n/(\log(2^{(\log n)^2}n^2))}\\ &\leq \frac{\delta^2}{2}+\log ne^{-n/((\log n)^2+2\log n)}\\ &\leq \frac{\delta^2}{2}+\frac{25}{n}\\ &\leq \frac{\delta}{4}.\end{aligned}$$

where last inequality follows from that $200/n<\delta<1/4$.

Total number of queries is

$$\begin{aligned}&\frac{1}{\varepsilon^2}O\left(n\log\frac{1}{\delta}+\sum_{i=\log\frac{1}{\delta}}^{\log n}\left(l_i+k_i+n\sqrt{\frac{n}{l_ik_i}}\right)\left(\log\frac{l_{i+1}}{\delta}\right)\right)\\ =&\frac{1}{\varepsilon^2}O\left(n\log\frac{1}{\delta}+\sum_{i=\log\frac{1}{\delta}}^{\log n-1}\left(i^6+\frac{n(i^2+\log\frac{1}{\delta})^2}{i^6}+n/(i^2+\log\frac{1}{\delta})\right)\log l_{i+1}\right)\\ =&\frac{1}{\varepsilon^2}O\left(n\log\frac{1}{\delta}+\sum_{i=\log\frac{1}{\delta}}^{\log n-1}\left(\left(i^6+\frac{n}{i^2}+\frac{n}{i^2}\right)(\log l_{i+1})^3\right)\right)\\ =&\frac{1}{\varepsilon^2}O\left(n\log\frac{1}{\delta}+\sum_{i=\log\frac{1}{\delta}}^{\log n-1}\left(\left(i^6+\frac{n}{i^2}\right)(\log(i+1))^3\right)\right)\\ =&O\left(\frac{n}{\varepsilon^2}\log\frac{1}{\delta}\right).\end{aligned}$$

□

5.E Proofs for Section 5.3.2

5.E.1 Proof of Lemma 77

Proof. Proof is similar to proof of [Theorem 2, [FHO$^+$17]] which proves the correctness and comparison complexity of SOFT-SEQ-ELIM. One can easily adapt that proof for AGNOSTIC-SEQ (by replacing δ/n with $\delta/(2i^2)$). □

5.F Motivation and Proofs for SubSection 5.3.3

Similar to subsection 5.2.2, to motivate the algorithm, we first present a general framework for any sequential and n-agnostic maximization algorithm and give sufficient conditions for its correctness. Based on these sufficient conditions, we identify where AGNOSTIC-SEQ uses unnecessary comparisons and derive the optimal algorithm reducing the overhead of AGNOSTIC-SEQ.

5.F.1 General Framework

Consider a general sequential framework GENERAL-AGNOSTIC-SEQ that maintains an anchor element r_i. At ith instance of a new element arrival, GENERAL-AGNOSTIC-SEQ compares r_{i-1} with $S(i)$ and updates the next anchor r_i using ANCHOR-UPDATE.

Notice that GENERAL-AGNOSTIC-SEQ is n-agnostic since ANCHOR-UPDATE never uses knowledge of n. n is used in lines 4 and 7 only for ease of notation. GENERAL-AGNOSTIC-SEQ runs as long as elements are presented, and outputs once stream of elements ends.

Now we state the two properties that if ANCHOR-UPDATE satisfies them, the correctness of GENERAL-AGNOSTIC-SEQ is guaranteed.

Algorithm 43 GENERAL-AGNOSTIC-SEQ

inputs
  Set S, bias ε, confidence δ
$r_1 \leftarrow S(1)$
for $i = 2$ to n **do**
  $r_i \leftarrow$ ANCHOR-UPDATE$(S(i), r_{i-1}, \varepsilon, \delta, i)$
end for
return r_n

5.F.2 Good Anchor Update

Consider ANCHOR-UPDATE that satisfies the following two properties:

- W.p. $\geq 1-\delta_i$, $\tilde{p}_{\text{ANCHOR-UPDATE}(e,f,\varepsilon,\delta,i),f} \geq 0$ s.t. $\sum_i \delta_i \leq \delta/4$
- W.p. $1-\delta/4$, $\tilde{p}_{\text{ANCHOR-UPDATE}(e,f,\varepsilon,\delta,i),e} \geq -\varepsilon$.

Put into words, ANCHOR-UPDATE$(e,f,\varepsilon,\delta,i)$, w.p.$\geq 1-\delta_i$ is preferable to f and w.p.$\geq 1-\delta/4$, is ε-preferable to e. Further $\sum_i \delta_i \leq \delta/4$. We call such ANCHOR-UPDATE a *good-anchor-update*.

In Lemma 81, we show that if ANCHOR-UPDATE is a *good-anchor-update* then w.p.$\geq 1-\delta/2$, GENERAL-AGNOSTIC-SEQ(S,ε,δ) returns an ε-maximum. Proof in Appendix 5.F.5.

Lemma 81. *If* ANCHOR-UPDATE *is a* good-anchor-update *algorithm, then w.p.*$\geq 1-\delta/2$, GENERAL-AGNOSTIC-SEQ(S,ε,δ) *returns an* ε*-maximum.*

Notice that ANCHOR-UPDATE$(e,f,\varepsilon,\delta,i) =$ COMPARE2$(e,f,0,\varepsilon,\delta/(8i^2))$ is a *good-anchor-update* and results in an algorithm that is essentially same as AGNOSTIC-SEQ. But using COMPARE2 with confidence parameter $\delta/(8i^2)$ is overkill since $\tilde{p}_{\text{COMPARE2}(e,f,0,\varepsilon,\frac{\delta}{8i^2}),e} \geq -\varepsilon$ with probability $1-\delta/(8i^2)$ which is much higher than the sufficient $1-\delta/4$.

5.F.3 OPT-ANCHOR-UPDATE

We now present an alternative of using COMPARE2 in one shot. Within each instance of ANCHOR-UPDATE, we use multiple rounds of COMPARE2, decreasing the confidence parameter

with each consecutive round such that overall comparisons used are orderwise same as comparisons used in a single instance of COMPARE2 with confidence parameter $\Theta(\frac{\delta}{i^2})$. Within each instance of ANCHOR-UPDATE we move to the next COMPARE2 round only if all the previous rounds returned 2. This helps in terminating ANCHOR-UPDATE much earlier than if only one round of COMPARE2 is used.

Algorithm 44 OPT-ANCHOR-UPDATE

inputs
  element e, element f, bias ε, confidence δ, number i
Initialize: $t \leftarrow 0$, $a \leftarrow 2$
while $a = 2$ and $t < \max(2, \log\log_{\frac{1}{\delta}} i^2 + 1)$ **do**
  $a \leftarrow \text{COMPARE2}(e, f, 0, \varepsilon, \delta^{2^t+1}/8)$
  $t \leftarrow t + 1$
end while
if $a = 1$ **then**
  return f
else
  return e
end if

In Lemma 82, we show that OPT-ANCHOR-UPDATE is *good-anchor-update*. Proof in Appendix 5.F.6

Lemma 82. OPT-ANCHOR-UPDATE *is a* good-anchor-update *algorithm.*

We now bound the number of comparisons used by OPT-ANCHOR-UPDATE. Proof in Appendix 5.F.7.

Lemma 83. OPT-ANCHOR-UPDATE$(e, f, \varepsilon, \delta, i)$ *uses* $O\left(\frac{1}{\varepsilon^2}\log\frac{i}{\delta}\right)$ *comparisons.*

Notice that even in the worst case, OPT-ANCHOR-UPDATE$(e, f, \varepsilon, \delta, i)$ uses orderwise same comparisons as COMPARE2$(e, f, 0, \varepsilon, \delta/(8i^2))$. We later in Lemma 86 show that in fact when OPT-ANCHOR-UPDATE used in GENERAL-AGNOSTIC-SEQ, more often uses much fewer comparisons than this pessimistic bound.

5.F.4 OPT-AGNOSTIC-SEQ

We now present our main algorithm OPT-AGNOSTIC-SEQ that uses OPT-ANCHOR-UPDATE in GENERAL-AGNOSTIC-SEQ.

Algorithm 45 OPT-AGNOSTIC-SEQ

inputs
  Set S, bias ε, confidence δ
$r_1 \leftarrow S(1)$
for $i = 2$ to n **do**
  $r_i \leftarrow$ OPT-ANCHOR-UPDATE$(S(i), r_{i-1}, \varepsilon, \delta, i)$
end for
return r_n

Since OPT-ANCHOR-UPDATE is a *good-anchor-update*, w.p.$\geq 1-\delta/2$, OPT-AGNOSTIC-SEQ outputs an ε-maximum and hence Lemma 84.

Lemma 84. *W.p.*$\geq 1-\delta/2$, OPT-AGNOSTIC-SEQ(S,ε,δ) *outputs an* ε*-maximum.*

Proof. Proof follows from Lemmas 81 and 82. □

Notice that as part of Proof for Lemma 84, we use a property of *good-anchor-update* and show that w.p.$\geq 1-\delta/4$, anchor only gets better. In other words, w.p.$\geq 1-\delta/4$, $\forall i > j$, $\tilde{p}_{r_i,r_j} \geq 0$. The probability of this event is already absorbed in probability of correctness of OPT-AGNOSTIC-SEQ. Henceforth, in the analysis we assume that anchor always gets better.

We now bound the number of comparisons used by OPT-AGNOSTIC-SEQ. Since OPT-ANCHOR-UPDATE$(e,f,\varepsilon,\delta,i)$ uses $O\left(\frac{1}{\varepsilon^2}\log\frac{i}{\delta}\right)$ comparisons, Lemma 85 on comparison complexity of OPT-AGNOSTIC-SEQ follows.

Lemma 85. OPT-AGNOSTIC-SEQ(S,ε,δ) *uses* $O\left(\frac{n}{\varepsilon^2}\log\frac{n}{\delta}\right)$ *comparisons.*

Proof. Proof follows from Lemma 83 □

Notice that for $\delta < 50/n^{1/3}$, since $\log\frac{1}{\delta}$ and $\log\frac{n}{\delta}$ are of the same order, OPT-AGNOSTIC-SEQ(S,ε,δ) uses $O\left(\frac{n}{\varepsilon^2}\log\frac{1}{\delta}\right)$ comparisons. Henceforth, we assume

$\delta > 50/n^{1/3}$ and prove that w.p.$\geq 1-\delta/2$, OPT-AGNOSTIC-SEQ(S,ε,δ) uses $O\left(\frac{n}{\varepsilon^2}\log\frac{1}{\delta}\right)$ comparisons.

We prove this by showing that for most of the instances of a new element arrival, OPT-ANCHOR-UPDATE invokes very few COMPARE2 calls hence significantly reducing the comparison complexity.

In Lemma 86, we upper bound the number of times a particular round is reached. Proof in Appendix 5.F.8.

Lemma 86. *Over all instances of* OPT-ANCHOR-UPDATE *during* OPT-AGNOSTIC-SEQ(S,ε,δ), *w.p.*$\geq 1-e^{1-\sqrt{\frac{mn'}{4n}}}-e^{-n\sqrt{\frac{4n}{mn'}}}$, *the number of times the round* $\max(3,\lceil\log\log_{\frac{1}{\delta}} m^2\rceil+2)$ *is reached is* $\leq m+n'+n\sqrt{\frac{36n}{mn'}}$.

In Lemma 87, using the upper bound on number of times a round is visited over all instances of OPT-ANCHOR-UPDATE, we bound the comparisons used by OPT-AGNOSTIC-SEQ. Proof in Appendix 5.F.9.

Lemma 87. *For* $\delta \geq 50/n^{1/3}$, *w.p.*$\geq 1-\delta/2$, OPT-AGNOSTIC-SEQ(S,ε,δ) *uses* $O\left(\frac{n}{\varepsilon^2}\log\frac{1}{\delta}\right)$ *comparisons.*

Theorem 88. *[Same as Theorem 78] Under SST models, w.p.*$\geq 1-\delta$, OPT-AGNOSTIC-SEQ (S,ε,δ) *uses* $O\left(\frac{n}{\varepsilon^2}\log\frac{1}{\delta}\right)$ *comparisons and outputs an* $\varepsilon-$*maximum.*

Proof. [Proof of Theorem 78]

Proof follows from Lemmas 84, 85, and 87. □

5.F.5 Proof of Lemma 81

Proof. Since ANCHOR-UPDATE is a *good-anchor-update* algorithm, w.p. $\geq 1-\delta_i$, $\tilde{p}_{r_i,r_{i-1}} \geq 0$ and w.p. $\geq 1-\delta/4$,

$\tilde{p}_{r_i,S(i)} \geq -\varepsilon$.

Using union bound and that $\sum_i \delta_i \leq \delta/4$, w.p.$\geq 1-\delta/4$, $\forall i$ $\tilde{p}_{r_i,r_{i-1}} \geq 0$. Hence by SST, w.p.$\geq 1-\delta/4$, $\forall j > i$, $\tilde{p}_{r_j,r_i} \geq 0$.

WLOG, let the position of absolute maximum element be k i.e., $\tilde{p}_{S(k),S(j)} \geq 0 \; \forall j$. Notice that since ANCHOR-UPDATE is *good-anchor-update*, w.p.$\geq 1-\delta/4$, $\tilde{p}_{r_k,S(k)} \geq -\varepsilon$.

Hence using union bound, w.p.$\geq 1-\delta/2$,

$$\tilde{p}_{S(k),r_{|S|}} \leq \tilde{p}_{S(k),r_k} \leq \varepsilon.$$

Hence w.p.$\geq 1-\delta/2$, $r_{|S|}$ is an ε-maximum. □

5.F.6 Proof of Lemma 82

Proof. Let element g be the output if OPT-ANCHOR-UPDATE$(e,f,\varepsilon,\delta,i)$.

We first show that w.p.$\geq 1-\delta/4$, $\tilde{p}_{g,e} \geq -\varepsilon$. If $\tilde{p}_{f,e} \geq -\varepsilon$, notice that either $g = e$ or $g = f$ satisfy that $\tilde{p}_{g,e} \geq -\varepsilon$. Hence assume that $\tilde{p}_{f,e} < -\varepsilon$. In other words $\tilde{p}_{f,e} > \varepsilon$ and hence COMPARE2$(e,f,0,\varepsilon,\delta')$ will return 2 with probability $1-\delta'$. Hence by union bound, w.p.

$$\geq 1 - \sum_t \delta^{2^t+1}/8 \geq 1-\delta/4$$

, (where last inequality follows from $\delta < 1/2$) for all t, COMPARE2$(e,f,0,\varepsilon,\delta^{2^t+1}/8)$ will return 2. Hence w.p.$\geq 1-\delta/4$, $g = e$ and hence $\tilde{p}_{g,e} = 0 \geq -\varepsilon$.

Now we show that w.p.$\geq 1-\frac{\delta}{8i^2}$, $\tilde{p}_{g,f} \geq 0$. If $\tilde{p}_{e,f} \geq 0$, notice that $g = e$ or $g = f$ result in $\tilde{p}_{g,f} \geq 0$. Hence assume that $\tilde{p}_{e,f} < 0$. Observe that $g = e$ only if all runs of COMPARE2 output 2. Specifically even a run for a $t' > \log\log_{\frac{1}{\delta}} i^2$ COMPARE2$(e,f,0,\varepsilon,\delta^{2^{t'+1}}/8)$ should return 2. The probability that COMPARE2$(e,f,0,\varepsilon,\delta^{2^{t'+1}}/8)$ returns 2 is

$$\leq \delta^{2^{t'+1}}/8 \leq (\delta/8)\delta^{2^{\log\log_{1/\delta} i^2}} = \delta/(8i^2).$$

Therefore w.p.$\geq 1-\delta/(8i^2)$, $g=f$ and hence $\tilde{p}_{g,f}=0\geq 0$. Noting that $\sum_{i=1}^{\infty}\frac{\delta}{8i^2}\leq\delta/4$ completes the proof. □

5.F.7 Proof of Lemma 83

Proof. OPT-ANCHOR-UPDATE$(e,f,\varepsilon,\delta,i)$ calls COMPARE2$(e,f,0,\varepsilon,\delta^{2^t+1}/8)$ for $t=1$ to $t=\lceil\log\log_{1/\delta}i^2\rceil$. Hence total comparisons used is

$$\begin{aligned}\sum_{i=1}^{\max(1,\lceil\log\log_{1/\delta}i^2\rceil)} O\left(\frac{1}{\varepsilon^2}\log\frac{8}{\delta^{2^t+1}}\right) &= O\left(\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\sum_{i=1}^{\max(1,\lceil\log\log_{1/\delta}i^2\rceil)}(2^t+1)\right)\\ &= O\left(\frac{2^{1+\max(1,\log\log_{1/\delta}i^2)}}{\varepsilon^2}\log\frac{1}{\delta}\right)\\ &= O\left(\frac{2\max(1,\log_{1/\delta}i^2)}{\varepsilon^2}\log\frac{1}{\delta}\right)\\ &= O\left(\frac{1}{\varepsilon^2}\left(\log\frac{1}{\delta}+\log\frac{1}{i^2}\right)\right)\\ &= O\left(\frac{1}{\varepsilon^2}\log\frac{i}{\delta}\right).\end{aligned}$$

□

5.F.8 Proof of Lemma 86

Proof. We introduce some notation that helps us in bounding number of times a round is invoked.

Definition 89. *Let $t_{n'}$ denote the n'th ranked element according to underlying descending order ranking. Further let $c_{n',\varepsilon,\delta,m,\alpha}$ denote the rank of the best element e such that with probability $\geq\alpha$,* OPT-ANCHOR-UPDATE$(t_{n'},e,\varepsilon,\delta,m)$ *outputs $t_{n'}$. In other words, (by SST), for all $n''\leq n'$ and $n'''\geq c_{n',\varepsilon,\delta,m,\alpha}$, with probability $\geq\alpha$,* OPT-ANCHOR-UPDATE$(t_{n''},t_{n'''},\varepsilon,\delta,m)$ *outputs* 2 *and for all $n''\geq n'$ and $n'''<c_{n',\varepsilon,\delta,,m,\alpha}$, the probability that* OPT-ANCHOR-UPDATE$(t_{n''},t_{n'''},\varepsilon,\delta,m)$ *outputs* 2 *is $\leq\alpha$.*

We first lower bound the probability that anchor at mth instance of OPT-ANCHOR-UPDATE is better than some element.

Lemma 90. *For all m, n', n s.t., w.p $\geq 1 - e^{1-\sqrt{\frac{mn'}{4n}}}$,*

$$\tilde{p}_{r_m, t_{c_{n',\varepsilon,\delta,m,\sqrt{\frac{4n}{mn'}}}}} \geq 0.$$

Proof. By Chernoff bound, w.p. $\geq 1 - e^{\frac{mn'}{4n}}$, in first m elements there will be at least $\frac{mn'}{4n}$ elements from first n' ranked elements.

Let $e = r_{c_{n',\varepsilon,\delta,m,\sqrt{\frac{4n}{mn'}}}}$. Let f be an element in top n' ranked elements and g be worse than e. For all $i \leq m$, w.p.$\geq \sqrt{\frac{4n}{mn'}}$, OPT-ANCHOR-UPDATE$(f, g, \varepsilon, \delta, i)$ outputs f. Hence, for any anchor worse than e, when compared with any one of the elements in the top n' ranked elements in an instance less than m gets replaced with probability $\geq \sqrt{\frac{4n}{mn'}}$. Since there are at least $\frac{mn'}{4n}$ such (top n' ranked elements) in instances less than m, the probability that an anchor worse than e not getting replaced by one of top $\frac{mn'}{4n}$ ranked elements is

$$\leq \left(1 - \frac{mn'}{4n}\right)^{\sqrt{\frac{4n}{mn'}}} \leq e^{-\sqrt{\frac{mn'}{4n}}}.$$

Proof follows from the union bound. □

From Lemma 90, w.p.$\geq 1 - e^{1-\sqrt{\frac{mn'}{4n}}}$,

$$\tilde{p}_{r_m, t_{c_{n',\varepsilon,\delta,m,\sqrt{\frac{4n}{mn'}}}}} \geq 0.$$

From now, we assume that this event has happened. Since r_m is better than $t_{c_{n',\varepsilon,\delta,m,\sqrt{\frac{4n}{mn'}}}}$ and anchor only gets better, probability that for $k > n'$, OPT-ANCHOR-UPDATE$(t_k, r_{m'}, \varepsilon, \delta, m')$ (for some

$m' > m$) reaches more than $\max(3, \lceil \log\log_{\frac{1}{\delta}} m^2 \rceil + 2)$ rounds is $\leq \sqrt{\frac{4n}{mn'}}$. Hence, by chernoff bound, after time m, out of all elements ranked outside top n', the number of elements that reach more than $\max(3, \lceil \log\log_{\frac{1}{\delta}} m^2 \rceil + 2)$ rounds is $\leq e^{-n\sqrt{\frac{4n}{mn'}}}$. Proof follows from union bound. □

5.F.9 Proof of Lemma 87

Proof. To prove the Lemma, consider the sequence of m_i and n'_i values.

$$m_i = i^6$$
$$n'_i = \frac{4n\left(\log\left(\frac{2^{i^2}}{\delta^2}\right)\right)^2}{m_i},$$

for $\log\frac{1}{\delta} \leq i \leq \log n$.

$$m_{\log n+1} = n.$$

From Lemma 86, w.p.$\geq 1 - e^{1-\sqrt{\frac{m_i n'_i}{4n}}} - e^{-n\sqrt{\frac{4n}{m_i n'_i}}}$, number of elements that are compared $\Omega(\frac{1}{\varepsilon^2}\log\frac{m_i}{\delta})$ times is $O\left(m_i + n'_i + n\sqrt{\frac{4n}{m_i n'_i}}\right)$. Hence by union bound, $\geq 1 - \sum_i (e^{1-\sqrt{\frac{m_i n'_i}{4n}}} - e^{-n\sqrt{\frac{4n}{m_i n'_i}}})$, and using upper sum, the total number of comparisons is

$$\frac{1}{\varepsilon^2} O\left(n\log\frac{1}{\delta} + \sum_{i=\log\frac{1}{\delta}}^{\log n}\left(m_i + n'_i + n\sqrt{\frac{n}{m_i n'_i}}\right)\left(\log\frac{m_{i+1}}{\delta}\right)\right).$$

We first bound the probability of event,

$$\begin{aligned}
\sum_{i=\log\frac{1}{\delta}}^{\log n}(e^{1-\sqrt{\frac{m_i n_i'}{4n}}}+e^{-n\sqrt{\frac{4n}{m_i n_i'}}}) &= \sum_{i=\log\frac{1}{\delta}}^{\log n}(e^{1-\log\frac{2^{i^2}}{\delta^2}}+e^{-n/(\log\frac{2^{i^2}}{\delta^2})} \\
&\leq \sum_{i=\log\frac{1}{\delta}}^{\log n} e\frac{\delta^2}{2^{i^2}}+e^{-n/(\log(2^{(\log n)^2}n^2))} \\
&\leq \frac{\delta^2}{2}+\log n e^{-n/((\log n)^2+2\log n)} \\
&\leq \frac{\delta^2}{2}+\frac{25}{n} \\
&\leq \frac{\delta}{2}.
\end{aligned}$$

where last inequality follows from that $50/n^{1/3} < \delta < 1/2$.

Total number of comparisons is

$$\begin{aligned}
&\frac{1}{\varepsilon^2}O\left(n\log\frac{1}{\delta}+\sum_{i=\log\frac{1}{\delta}}^{\log n}\left(m_i+n_i'+n\sqrt{\frac{n}{m_i n_i'}}\right)\left(\log\frac{m_{i+1}}{\delta}\right)\right) \\
=&\frac{1}{\varepsilon^2}O\left(n\log\frac{1}{\delta}+\sum_{i=\log\frac{1}{\delta}}^{\log n-1}\left(i^6+\frac{n(i^2+\log\frac{1}{\delta})^2}{i^6}+n/(i^2+\log\frac{1}{\delta})\right)\log m_{i+1}\right) \\
=&\frac{1}{\varepsilon^2}O\left(n\log\frac{1}{\delta}+\sum_{i=\log\frac{1}{\delta}}^{\log n-1}\left(\left(i^6+\frac{n}{i^2}+\frac{n}{i^2}\right)(\log m_{i+1})^3\right)\right) \\
=&\frac{1}{\varepsilon^2}O\left(n\log\frac{1}{\delta}+\sum_{i=\log\frac{1}{\delta}}^{\log n-1}\left(\left(i^6+\frac{n}{i^2}\right)(\log(i+1))^3\right)\right) \\
=&O\left(\frac{n}{\varepsilon^2}\log\frac{1}{\delta}\right).
\end{aligned}$$

□

5.G Proofs for Section 5.4

5.G.1 Proof of Theorem 79

Proof. Proof is very similar to Proof of [Theorem 3 [SG19]]. We adapt the proof to show that the same proof still works for more general setting that satisfies both SST and STI properties.

We first show that w.p.$\geq 1-\delta/30$, best item is never eliminated and hence w.p.$\geq 1-\delta/15$, final output is indeed an ε-maximum.

Lemma 91. *W.p.$\geq 1-\delta/30$, best item is never eliminated.*

Proof. Notice that an element e is eliminated only if COMPARE2$(r,e,0,\varepsilon_i,\delta_i)$ returns 2. Notice that for best element b since $\tilde{p}_{b,f} \geq 0\ \forall f$. By Lemma 76, w.p.$\geq 1-\delta_i$, for any e, COMPARE2$(e,b,0,\varepsilon_i,\delta_i)$ outputs 1. Hence by union bound, w.p.$\geq 1-\sum_i \delta_i \geq 1-\delta/30$, for all i, COMPARE2$(r_i,b,0,\varepsilon_i,\delta_i)$ output 1 and therefore best element is never eliminated. □

In the Lemma below, we prove the correctness of COMPETITIVE-MAX.

Lemma 92. *W.p.$\geq 1-\delta/15$,* COMPETITIVE-MAX(S,ε,δ) *outputs an ε-maximum.*

Proof. From Lemma 91 w.p.$\geq 1-\delta/30$, best element is never eliminated. Let o be output when PAC-MAX$(S,\varepsilon,\delta/30)$ is called in the end. Since S still contains the best element b, w.p.$\geq 1-\delta/30$, $\tilde{p}_{b,o} \leq \varepsilon$. Hence by union bound, $\geq 1-\delta/15$, the final output will be an ε-maximum. □

Similar to [SG19], we define a set of elements $T_z = \{e : \frac{1}{2^z} \leq \tilde{p}_{b,e} \leq \frac{1}{2^{z-1}}\}$. Let the set of elements remaining after s instances of PAC-MAX be A_s and $A_{z,s} = A_s \bigcap T_z$.

Lemma 93. *Assuming the best arm is never eliminated, w.p.$\geq 1-\frac{\delta}{10}$, for all $s \geq z$, $|A_{z,s+1}| \leq \frac{2}{19}|A_{z,s}|$.*

Proof. Since best arm is never eliminated, using union bound w.p.$\geq 1-\delta/30$, PAC-MAX always outputs an ε_i-maximum. Let the output of sth instance of PAC-MAX be r_s. r_s is an $1/2^{s+1}$-maximum of S. Hence by SST and STI (only time STI is used), for any $f \in A_{z,s}$, $\tilde{p}_{r_s,f} \geq 1/2^{s+1}$. Hence COMPARE2$(r_s, f, 0, 1/2^{s+1}, \delta_s)$ outputs 2 with probability $1-\delta_s$. Hence $\mathbb{E}|A_{z,s+1}| \leq \delta_s \mathbb{E}|A_{z,s}|$. Hence by Markov's inequality,

$$Pr(|A_{z,s+1}| \leq \frac{2}{19}|A_{z,s}|) \leq 19\delta_s/2.$$

Finally, applying union bound over all s and $z \leq s$, the Lemma follows. □

This completes the probabilistic comaprisons part and from here Lemmas 15 and 16 of [SG19] bounds the comparison complexity using Lemmas 91 and 93. □

Bibliography

[AB10] Jean-Yves Audibert and Sébastien Bubeck. Best arm identification in multi-armed bandits. In *COLT-23th Conference on learning theory-2010*, pages 13–p, 2010.

[AFHN15] Miklós Ajtai, Vitaly Feldman, Avinatan Hassidim, and Jelani Nelson. Sorting and selection with imprecise comparisons. *ACM Transactions on Algorithms (TALG)*, 12(2):19, 2015.

[AFJ+16] Jayadev Acharya, Moein Falahatgar, Ashkan Jafarpour, Alon Orlitsky, and Ananda Theertha Suresh. Maximum selection and sorting with adversarial comparators and an application to density estimation. *arXiv preprint arXiv:1606.02786*, 2016.

[AJOS14a] Jayadev Acharya, Ashkan Jafarpour, Alon Orlitsky, and Ananda Theertha Suresh. Near-optimal-sample estimators for spherical gaussian mixtures. *NIPS*, 2014.

[AJOS14b] Jayadev Acharya, Ashkan Jafarpour, Alon Orlitsky, and Ananda Theertha Suresh. Sorting with adversarial comparators and application to density estimation. In *ISIT*, pages 1682–1686. IEEE, 2014.

[BFHS14] Róbert Busa-Fekete, Eyke Hüllermeier, and Balázs Szörényi. Preference-based rank elicitation using statistical models: The case of mallows. In *Proc. of the ICML*, pages 1071–1079, 2014.

[BFSH14] Róbert Busa-Fekete, Balázs Szörényi, and Eyke Hüllermeier. Pac rank elicitation through adaptive sampling of stochastic pairwise preferences. In *AAAI*, 2014.

[BMR10] Linas Baltrunas, Tadas Makcinskas, and Francesco Ricci. Group recommendations with rank aggregation and collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 119–126. ACM, 2010.

[BMS09] Sébastien Bubeck, Rémi Munos, and Gilles Stoltz. Pure exploration in multi-armed bandits problems. In *Algorithmic Learning Theory*, pages 23–37. Springer, 2009.

[BT52] Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

[C+15] Sourav Chatterjee et al. Matrix estimation by universal singular value thresholding. *The Annals of Statistics*, 43(1):177–214, 2015.

[CBCTH13] Xi Chen, Paul N Bennett, Kevyn Collins-Thompson, and Eric Horvitz. Pairwise ranking aggregation in a crowdsourced setting. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 193–202. ACM, 2013.

[CN91] Andrew Caplin and Barry Nalebuff. Aggregation and social choice: a mean voter theorem. *Econometrica: Journal of the Econometric Society*, pages 1–23, 1991.

[DHS+15] Miroslav Dudík, Katja Hofmann, Robert E Schapire, Aleksandrs Slivkins, and Masrour Zoghi. Contextual dueling bandits. *arXiv preprint arXiv:1502.06362*, 2015.

[EDMM02] Eyal Even-Dar, Shie Mannor, and Yishay Mansour. Pac bounds for multi-armed bandit and markov decision processes. In *Computational Learning Theory*, pages 255–270. Springer, 2002.

[EDMM06] Eyal Even-Dar, Shie Mannor, and Yishay Mansour. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of machine learning research*, 7(Jun):1079–1105, 2006.

[FHO+17] Moein Falahatgar, Yi Hao, Alon Orlitsky, Venkatadheeraj Pichapati, and Vaishakh Ravindrakumar. Maxing and ranking with few assumptions. In *Advances in Neural Information Processing Systems*, pages 7063–7073, 2017.

[FJO+18] Moein Falahatgar, Ayush Jain, Alon Orlitsky, Venkatadheeraj Pichapati, and Vaishakh Ravindrakumar. The limits of maxing, ranking, and preference learning. In *International Conference on Machine Learning*, pages 1426–1435, 2018.

[FOPS17] Moein Falahatgar, Alon Orlitsky, Venkatadheeraj Pichapati, and Ananda Theertha Suresh. Maximum selection and ranking under noisy comparisons. In *International Conference on Machine Learning*, pages 1088–1096, 2017.

[Fre83] PR Freeman. The secretary problem and its extensions: A review. 1983.

[FRPU94] Uriel Feige, Prabhakar Raghavan, David Peleg, and Eli Upfal. Computing with noisy information. *SIAM Journal on Computing*, 23(5):1001–1018, 1994.

[GGL12] Victor Gabillon, Mohammad Ghavamzadeh, and Alessandro Lazaric. Best arm identification: A unified approach to fixed budget and fixed confidence. In *Advances in Neural Information Processing Systems*, pages 3212–3220, 2012.

[gif] `http://www.gif.gf/`.

[GM06] John P Gilbert and Frederick Mosteller. Recognizing the maximum of a sequence. In *Selected Papers of Frederick Mosteller*, pages 355–398. Springer, 2006.

[HFCB08] Eyke Hüllermeier, Johannes Fürnkranz, Weiwei Cheng, and Klaus Brinker. Label ranking by learning pairwise preferences. *Artificial Intelligence*, 172(16-17):1897–1916, 2008.

[HMG06] Ralf Herbrich, Tom Minka, and Thore Graepel. Trueskill: a bayesian skill rating system. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, pages 569–576. MIT Press, 2006.

[HSRW16] Reinhard Heckel, Nihar B Shah, Kannan Ramchandran, and Martin J Wainwright. Active ranking from pairwise comparisons and when parametric assumptions don't help. *arXiv preprint arXiv:1606.08842*, 2016.

[JKDN15] Kevin Jamieson, Sumeet Katariya, Atul Deshpande, and Robert Nowak. Sparse dueling bandits. In *Artificial Intelligence and Statistics*, pages 416–424, 2015.

[JKSO16] Minje Jang, Sunghyun Kim, Changho Suh, and Sewoong Oh. Top-k ranking from pairwise comparisons: When spectral ranking is optimal. *arXiv preprint arXiv:1603.04153*, 2016.

[KKS13] Zohar Karnin, Tomer Koren, and Oren Somekh. Almost optimal exploration in multi-armed bandits. In *International Conference on Machine Learning*, pages 1238–1246, 2013.

[LGAL14] David Timothy Lee, Ashish Goel, Tanja Aitamurto, and Helene Landemore. Crowdsourcing for participatory democracies: Efficient elicitation of social choice functions. In *Second AAAI Conference on Human Computation and Crowdsourcing*, 2014.

[Lin61] Denis V Lindley. Dynamic programming and decision theory. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 10(1):39–51, 1961.

[Luc05] R Duncan Luce. *Individual choice behavior: A theoretical analysis*. Courier Corporation, 2005.

[Mal57] Colin L Mallows. Non-null ranking models. i. *Biometrika*, 44(1/2):114–130, 1957.

[MSE17] Soheil Mohajer, Changho Suh, and Adel Elmahdy. Active learning for top-k rank aggregation from noisy comparisons. In *International Conference on Machine Learning*, pages 2488–2497, 2017.

[Muk11] Sudipta Mukherjee. *Data structures using C: 1000 problems and solutions*. McGraw Hill Education, 2011.

[NOS12] Sahand Negahban, Sewoong Oh, and Devavrat Shah. Iterative ranking from pairwise comparisons. In *NIPS*, pages 2474–2482, 2012.

[NOS16] Sahand Negahban, Sewoong Oh, and Devavrat Shah. Rank centrality: Ranking from pairwise comparisons. *Operations Research*, 2016.

[Pla75] Robin L Plackett. The analysis of permutations. *Applied Statistics*, pages 193–202, 1975.

[RA14] Arun Rajkumar and Shivani Agarwal. A statistical convergence perspective of algorithms for rank aggregation from pairwise data. In *Proc. of the ICML*, pages 118–126, 2014.

[RJ07] Filip Radlinski and Thorsten Joachims. Active exploration for learning rankings from clickthrough data. In *Proceedings of the 13th ACM SIGKDD*, pages 570–579. ACM, 2007.

[RKJ08] Filip Radlinski, Madhu Kurup, and Thorsten Joachims. How does clickthrough data reflect retrieval quality? In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 43–52. ACM, 2008.

[Rob52] Herbert Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5):527–535, 1952.

[RS77] H Rubin and SM Samuels. The finite-memory secretary problem. *The Annals of Probability*, pages 627–635, 1977.

[SBFPH15] Balázs Szörényi, Róbert Busa-Fekete, Adil Paul, and Eyke Hüllermeier. Online rank elicitation for plackett-luce: A dueling bandits approach. In *NIPS*, pages 604–612, 2015.

[SBGW16] Nihar Shah, Sivaraman Balakrishnan, Aditya Guntuboyina, and Martin Wainwright. Stochastically transitive models for pairwise comparisons: Statistical and computational issues. In *International Conference on Machine Learning*, pages 11–20, 2016.

[SBW16] Nihar B Shah, Sivaraman Balakrishnan, and Martin J Wainwright. Feeling the bern: Adaptive estimators for bernoulli probabilities of pairwise comparisons. *arXiv preprint arXiv:1603.06881*, 2016.

[SCPX13] Hossein Azari Soufiani, William Chen, David C Parkes, and Lirong Xia. Generalized method-of-moments for rank aggregation. In *Advances in Neural Information Processing Systems*, pages 2706–2714, 2013.

[SG19] Aadirupa Saha and Aditya Gopalan. From pac to instance-optimal sample complexity in the plackett-luce model. *arXiv preprint arXiv:1903.00558*, 2019.

[Sko10] Michal Skorepa. *Decision making: a behavioral economic approach*. Palgrave Macmillan, 2010.

[SKS16] Vasilis Syrgkanis, Akshay Krishnamurthy, and Robert E Schapire. Efficient algorithms for adversarial contextual learning. *arXiv preprint arXiv:1602.02454*, 2016.

[UCFN13] Tanguy Urvoy, Fabrice Clerot, Raphael Féraud, and Sami Naamane. Generic exploration and k-armed voting bandits. In *Proc. of the ICML*, pages 91–99, 2013.

[Wik17] Wikipedia. Nontransitive dice — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Nontransitive%20dice&oldid=779713322`, 2017. [Online; accessed 19-May-2017].

[WSE14] Jialei Wang, Nathan Srebro, and James Evans. Active collaborative permutation learning. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 502–511. ACM, 2014.

[YBKJ12] Yisong Yue, Josef Broder, Robert Kleinberg, and Thorsten Joachims. The k-armed dueling bandits problem. *Journal of Computer and System Sciences*, 78(5):1538–1556, 2012.

[YJ11] Yisong Yue and Thorsten Joachims. Beat the mean bandit. In *Proc. of the ICML*, pages 241–248, 2011.

[ZC14] Yuan Zhou and Xi Chen. Optimal pac multiple arm identification with applications to crowdsourcing. 2014.

[ZCL14] Yuan Zhou, Xi Chen, and Jian Li. Optimal pac multiple arm identification with applications to crowdsourcing. In *International Conference on Machine Learning*, pages 217–225, 2014.